January/February 1982

Volume 2, Number 1

$2.50 (USA)
£1.20 (UK)

The magazine for Sinclair users

SYNC®

Machine Code:
- Game inventing
- Life in MC
- Window

Math:
- Floating points
- Linear regression

Games and Programs:
- Lunar Landing
- Battleship Solitaire
- ZX80 Cipher Machine
- Tioga Toads
- Dice and Train

SYNTAX ZX80®

A PUBLICATION OF **THE HARVARD GROUP**

SYNTAX ZX80 is a monthly newsletter exclusively for ZX80, ZX81 and MicroAce owners. We bring you news, reviews and applications for your computer, plus technical notes for circuit-builders. SYNTAX also provides a forum for thousands of users to share advice and problems about programs and vendors. We bring you timely updates about new hardware, software and books. And we cover *all* the Sinclair-MicroAce computers, including the new ZX81.

At SYNTAX we emphasize practicality. You can apply our suggestions even if you aren't sure at first why they work, because we give you complete instructions. Text is clear and easy to understand. SYNTAX readers already know about:

- An automatic phone-dialer they can put together in a few hours
- Syntactic Sums™ to check input for errors
- Printing characters four times normal size
- Programs to explore computer memory
- Cassette eavesdropping to locate files on tape and simplify loading
- How to build their own external additional RAM
- How to add an 8212 I/O chip to control external devices from their computers

And SYNTAX readers like what they get every month. Subscribers know they can depend on us.

> *After receiving only three issues of SYNTAX ZX80, I find that I anxiously await the next issue . . . keep up the good work!*
>
> Martin Irons
> Goshen, NY

> *Congratulations on the brass-tacks, down-to-earth approach of your newsletter. I'll be looking forward to future issues.*
>
> Otis Imboden
> Washington, DC

Many readers get their first issue and immediately order the back issues — more proof that they like what they see.

What's special about our publication? Just look through one issue. We work hard to bring you a quality newsletter. We strive to print useful programs of above-average accuracy. As any computer magazine editor can tell you, program listing accuracy is tough to achieve, but we boost our average with every issue. We test each program to make sure it works, it fits in the designated RAM, and it runs when you follow the directions. We print program listings in screen-image format to make it easier for you (it's sure not easier for us!) to enter programs accurately. We invented Syntactic Sum™ as an additional aid for you in getting error-free programs. With your subscription you also get access to thousands of other readers, and our staff experts are available by phone to answer your questions or help you solve problems with your machine.

SYNTAX readers get every month:

- Latest news of Z80 hardware and software
- Programs to organize information, calculate, entertain, or instruct
- Do-it-yourself additions to the ZX80/MicroAce/ZX81
- Clear explanations for beginners

To share the benefits of SYNTAX ZX80, just complete the coupon below and return it with your choice of payment. You will receive a year's subscription, 12 issues, for only $29 in US funds (plus $14 for foreign airmail if you live outside North America).

We are so sure you'll find SYNTAX useful that we promise to refund your entire subscription fee if you aren't satisfied. An unconditional guarantee — you can't lose. But if you're still skeptical, send $1 for a sample issue and see for yourself how SYNTAX can help you use and enjoy your ZX80 or ZX81 more.

Join the others who stretch their ZX80s and ZX81s to the utmost. Act now — as soon as we receive your coupon with payment, your first issue will be on its way. For faster service, phone your credit card order to 617/456-3661. Don't miss SYNTAX!

THE
HARVARD
GROUP

RD 2 BOX 457
HARVARD, MA 01451

I own a ☐ Sinclair ZX80
☐ Sinclair ZX81
☐ MicroAce computer.

YES! Please send me 12 issues of SYNTAX for $29.

☐ My check for $29 is enclosed.
Make checks payable to:
SYNTAX ZX80, INC.

☐ Please charge my ☐ MasterCard
☐ VISA ☐ American Express
☐ Diner's Club account.

account number ______________________________

exp. date ________________ bank number (MC only) ________________

signature ______________________________

Name ________________ Title ________________

Organization ______________________________

Address ______________________________

City ________________ State ________ Zip ________

Day Phone (   ) ________________ Evening Phone (   ) ________________

S182

The Sinclair ZX-81 is innovative and powerful. Now there's a magazine to help you get the most out of it.

GET IN SYNC

Thousands of smart consumers have picked the Sinclair ZX-81 as their personal computer. And, unlike many of today's bargains, this one can really give you your money's worth. Or it can turn into nothing but an expensive calculator. A Sinclair owner can putter along in first gear, missing the power and potential of the ZX-81, or he can shift into high, pushing the ZX-81 beyond imaginable limits. That's why thousands of smart consumers have picked **SYNC** as their computer magazine.

Right on Target

The ZX-81 is unique. There is nothing like it, nothing that comes close to packing so much power and versatility into one small package. Some computer magazines might publish one or two articles about the Sinclair each year, some never mention it. **SYNC** covers only the ZX-81 and its predecessor, the ZX-80. If an article doesn't apply to the Sinclair, if a game doesn't work on the Sinclair, you won't see it in **SYNC**. Our staff and contributors are Sinclair owners. Some started out as experts. Others started as readers and became experts.

How can a whole magazine find enough material about one small computer? By covering everything from hardware to software, by offering both new applications and old tricks with a new twist. Did you know that the Sinclair can generate music? Our readers found that out when we published a program and article showing how to do it, and explaining why it works. Do you know where to buy software, books, or peripherals for the ZX-81? We list resources in every issue, along with addresses for user's groups so you can get in touch with other Sinclair owners. But knowing where to buy is not enough by a long shot. And that's where we can really help you out.

Hard-Hitting Evaluations

As a Sinclair owner, you know the value of a dollar. But it isn't always easy to know the value of all the extras on the market. Face it, some programs are great, some aren't worth the tape they're stored on. We receive every new product for the Sinclair as soon as it is available, often months before it is on the market. And those products are reviewed and tested with a very critical eye. If an advertiser doesn't care for this sort of honesty, we don't care for his business. We haven't gotten where we are by patting backs, we've gotten there by giving the Sinclair owner the information he needs. But there's more to **SYNC** than just reviews.

Applications and Explanations

The ZX-81 comes with a very powerful Basic language. But power doesn't imply difficulty. We show you how to get the most from your computer, whether you want to write a game or keep track of a mailing list. And we don't stop with Basic. The Sinclair can be programmed in machine language. For the newcomer, we have articles explaining machine language from the ground up. For the old pro (and anyone who has been reading **SYNC** for a while will soon find himself in this category) we have sophisticated routines for animation, data handling, and every other aspect of programming.

Don't run your computer in first gear.

Topping if off, hardware articles cover everything from attaching a full-size keyboard to adding a tape monitor. Whether you are interested in software or soldering, we'll keep you busy. But we also know how to have fun.

Games of Every Kind

If you like to shoot down attacking spaceships, fight monsters in a dungeon, or land on the moon, we've got what you want. Every issue of **SYNC** is packed with games. There are classic computer games converted for the Sinclair, and new games designed specifically to exploit the capabilities of the ZX-81. Our contributors keep getting better and better, but that's not surprising, because the games come complete with tips and explanations. Programming tricks and special techniques are fully explained, so you can use them in your own games. We don't believe in keeping secrets.

SYNC is a Creative Computing publication. **Creative Computing** is the number 1 magazine of software and applications with over 150,000 circulation. The two most popular computer games books in the world, *Basic Computer Games* and *More Basic Computer Games* (combined sales over 500,000) are published by Creative Computing. Creative Computing Software manufactures over 150 software packages for six different personal computers.

Order SYNC Today and Save Money!

SYNC
P.O. Box 789-M
Morristown, N.J. 07960

YES! Send me *SYNC* for the term checked:

☐ One year (6 issues) $12.97—I save 19%!

☐ Two years (12 issues) $22.97—I save 28%!

☐ Three years (18 issues) $31.97—I save 33%!

Savings based on full one-year subscription price of $16.

CHECK ONE:

☐ Payment enclosed.

☐ Bill me later.

Mr.
Mrs.
Ms.____________________________
(please print full name)

Address______________________Apt.____

City____________________________

State______________Zip____________

Foreign postage: Add $3 a year for Canada. Add $5 a year (cash payment in U.S. currency only) for all other countries outside U.S. and possessions. Please allow 60 to 90 days for delivery of first issue.

SPAJ

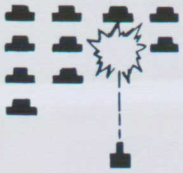
ZX81 SOFTWARE: A TO ZZZAPPP!!

WALLBUSTERS, CHESS, SUPER INVASION, ADVENTURE AND MORE...

SUPER INVASION

"The best Sinclair game to hit the market." —SYNC Magazine. A moving graphics game with three levels of play. SUPER INVASION challenges your skill as you fire lasers at the attacking space invaders while maneuvering your space craft to avoid their deadly lasers.

1K $14.95



WALLBUSTERS

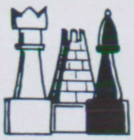
"A breakthrough in creating active display games."-SYNC Magazine. WALLBUSTERS challenges you to break through two barricades using nine balls and a curved bat. With seven levels of play, WALLBUSTERS is hard to beat. You'll be amazed at the superb graphics in this 1K game.

1K $14.95

ZXCHESS

This sophisticated chess game has seven levels of play and a detailed display of the board. You can change sides and even change levels of difficulty during a game. You can also start playing from any point in the game and if you get stuck, the computer will recommend a move.

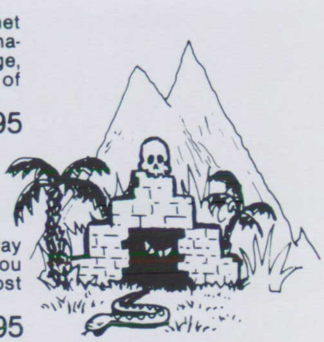
16K $24.95



ADVENTURE "A"

Your space ship is marooned on a strange planet but you can get out if you make the right combination of decisions. Written in machine language, this challenging adventure has over 100 words of vocabulary.

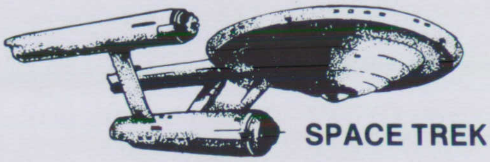
16K $19.95



ADVENTURE "B"

Enter the long lost Inca Temple, find your way through the tricky tunnels and corridors and you may find the lost treasure. Or you may be lost forever.

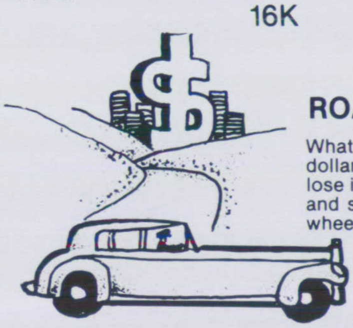
16K $19.95



SPACE TREK

As commander of Starship Enterprise, you find yourself defending a galaxy overrun with the dreaded DRAKONS. Can you destroy them? With five levels of play and excellent graphics, you'll find SPACE TREK entertaining and challenging. Can only be used with the ZX81.

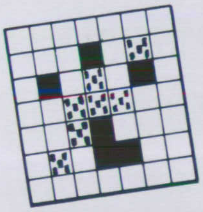
16K $14.95



ROAD TO RICHES

What would you do if someone gave you a million dollars to invest? Would you make more money or lose it all? This investment game combines luck and strategy to challenge up to four players to wheel and deal their way to riches...or ruin.

16K $14.95



REVERSI

if you like Othello, you'll love REVERSI. With the board displayed, you can go first or let the computer go and you have a choice of starting positions.

1K $14.95

TEN EXCITING PROGRAMS FOR THE ZX81: 1K

These BASIC programs on cassette include Lunar Lander, Space War and Brands Hatch. Included are complete listings and suggestions so you can learn and adapt programs. Can only be used with the ZX81.

1K $14.95

DIRECTORY/RECORD

Two programs on one cassette with full listings. The DIRECTORY program allows you to read names of programs saved on cassette. The RECORD program allows you to save and load 96 Byte data records on tape and can be incorporated into other programs.

1K $14.95

MACHINE LANGUAGE PROGRAMMING

Made Simple For the Sinclair ZX81. This book is a complete guide in machine language for the beginner.

BOOK $19.95

UNDERSTANDING YOUR ZX81 ROM

This book gives an overview of machine language and describes the operation of the Sinclair ROM. Essential for the serious programmer.

BOOK $19.95

COMING SOON:
PACKPERSON
AND MANY MORE ZX81
PROGRAMS! WRITE
FOR FREE CATALOG.

(Programs can also be used with the ZX80 with 8K ROM unless specified.)

ALL PROGRAMS ON CASSETTE	PRICE	QTY	AMOUNT
SUPER INVASION	14.95		
WALLBUSTERS	14.95		
ZXCHESS	24.95		
ADVENTURE "A"	19.95		
ADVENTURE "B"	19.95		
SPACE TREK	14.95		
REVERSI	14.95		
ROAD TO RICHES	14.95		
TEN EXCITING PROGRAMS	14.95		
DIRECTORY/RECORD	14.95		
MACHINE LANGUAGE	19.95		
UNDERSTANDING YOUR ROM	19.95		
SHIPPING AND HANDLING	1.50		$1.50
SALES TAX (NY RESIDENTS ONLY)			
	TOTAL:		

SEND CHECK OR MONEY ORDER TO:

SOFTSYNC, INC.
P.O. Box 480, Murray Hill Station, New York, N.Y. 10156

New York residents add sales tax.

NAME ____________________

ADDRESS ____________________

CITY ____________________

STATE/ZIP ____________________

The magazine for Sinclair users

SYNC®

January/February 1982

Volume 2, Number 1

Staff

Publisher/Editor-in-Chief	**David H. Ahl**
Managing Editor	**Paul Grosjean**
Associate Editor	**David Lubar**
Secretary	**Elizabeth Magin**
Production Manager	**Laura MacKenzie**
Art Director	**Susan Gendzwil**
Assistant Art Director	**Diana Negri**
Typesetters	**Jean Ann Vokoun** **Maureen Welsh**
Financial Coordinator	**William L. Baumann**
Personnel and Finance	**Patricia Kennelly**
Customer Service	**Ralph Loveys**
Circulation	**Frances Miskovich** **Carol Vita**

MEMBER
MITA

Index to Advertisers

Artic Computing	9
Be a Computer Literate	29
Books for the ZX81	cover 4
Burnett Electronics	11
Byte Back	13
Colossal Computer Cartoon Book	39
Computer Coin Games	39
Gladstone Electronics	7
Harvard Group	cover 2
JMC	27
JS&A	5
Katie and the Computer	29
Kopak Creations, Inc.	9
L.J.H. Enterprises	6
Marex	31
Melbourne House	cover 3
Mindware	6
Professional Electronics	27
RKL Systems	18
Sinclair Research	22,23
Softsync, Inc.	2
SYNC subscriptions	1

Volume 2, Number 1

SYNC (USPS: 585-490; ISSN: 0279-5701) is published bi-monthly for $16 per year by Creative Computing, 39 E. Hanover Ave., Morris Plains, NJ 07950. Second class postage paid at New York, NY 10016, and at additional mailing offices.

Subscription rates: USA: 6 issues $16; 12 issues $30; 18 issues $42. Canada: $3 per year additional. Other foreign: $5 per year additional. U.K. Air: 6 issues £13. Minimum charge card order $10.00.

U.K. address: *SYNC*, 27 Andrew Close, Stoke Golding, Nuneaton CV13 6EL.

Postmaster: Send address changed to SYNC, P.O. Box 789-M, Morristown, NJ 07960.

A
creative computing
PUBLICATION

letters

8K ROM Double Image

Dear Editor:

To answer partially Joe Sutton's question (*SYNC* 1:4) about the 8K ROM's "double image," I noticed the same thing with the 4K ROM: 4 "images" of the same 4K monitor program were to be found in the 16K addressing space set aside for system ROM. I reasoned that the unneeded most significant bits of the address bus were ignored, enabling the ROM chip whenever the address was below 16K. Thus, for the 8K ROM, address 0 and address 0 plus 8K are interpreted as the same address. The 8K bit is ignored. However, I do not understand the nitty-gritty hardware aspects fully (the ROM appears to have both the A8 and A12 address lines connected to chip select).

I would like to see an indepth discussion of the ZX80's discrete components (not much to discuss in the ZX81!). I would also like to see more articles on the 4K and 8K ROMs as well as hardware interfaces. For instance, I would like to purchase a surplus terminal (Model 15 or Model 33), but I do not know whether the standard Z80 interface will suffice or what modifications of the 8K ROM's LPRINT are needed.

As a math teacher I would like to see articles on educational applications since with tight money in government the inexpensive, but effective ZX81 may be the only realistic choice for today's schools.

I really have to compliment Mr. Sinclair on the improvements and reduced price of the ZX81. I only wish I would have purchased a ZX81 kit rather than an 8K ROM upgrade; then I would have two computers and smooth display for only $60 more. It would also be nice to have more on board RAM. So how about an article on constructing a memory upgrade using one of the many available RAM PC boards?

I would also like to see an article on constructing or interfacing a 2716/2732 EPROM programmer (student proof programs!

Timothy McIlwee
Granada Royale Rm. 121
Route 1, and I-95
Ormond Beach, FL 32074

Ed.—Here is a list of challenges for **SYNC** *authors. Similar requests have come from other readers.*

Taxman

Dear Editor:

The program *Taxman* that appears in *SYNC* (1:6) is an interesting mathematical game. The listing (as printed) does contain one minor bug. Line 630 should read

```
630 FOR I=3 TO N
```

to cover all possible cases. It would not hurt if, in fact, that line were to read

```
630 FOR I=1 TO N
```

As presently written, line 630 will not allow the computer to have credit for a remaining unused value of 3. To test this out, run the program for a set of 5 numbers.

It is not difficult to show that for a set of 30 numbers, the best score attainable is 301 to 164. The opening choices are 29, 25, 15, and 27. The remainder of the solution is left as an exercise for the reader. The first move in this game is obviously to pick the largest prime number in the set.

Peter D. Hoffman
5618 Martinique Dr.
Corpus Christi, TX 78411

An Inventory System

Dear Editor:

In reference to Dr. Stephen A. Justham's article entitled "An Inventory System," much space and some computing time can be saved in the search sequence by making the following changes:

```
   3 DIM R$(1,15)
3040 INPUT R$(1)
3042 delete
3044 delete
3052 delete
3068 IF R$(1)=I$(B) THEN GOTO 3100
3115 PRINT TAB 5; R$(1)
```

By setting up a single position array of 15 length at line 3, the input at line 3040 is automatically adjusted to equal the elements of the I$ array. This saves setting up the 151 position M$ array, resulting in a saving of up to 2,250 bytes of RAM.

Ralph Goodrick
3700 W. 151st
Stanley, KS 66224

Hampson's Plane

Dear Editor:

I feel "Hampson's Plane" is one of the better 4K/1K games you have published. I made a minor change to make continuous play easier. Add lines:

```
1CLS
105 IF K$="X" THEN GOTO 1
```

When one has completed a plane (or whenever an alpha coordinate is requested) an "X" can be entered to return to the beginning of the game. I also suggest that the string in line 20 be changed to one space. It makes the appearance of a plane similar to the one published in the article.

Robert Masters
396 Billerica Road
Tewksbury, MA 01876

ELECTRONIC FAIRY TALE

Remembering numbers is genie work.

Busy Buttons

Turn those innocent little buttons on your telephone into Busy Buttons and release the genie from its little black box.

A fairy tale? The story you are about to read may be true or it may be false.

If the story is false, we've wasted a lot of your time. If the story is true, well...you might just make a lucky discovery. Here's why.

Your push button telephone has a bunch of buttons that make beeping sounds when you press them. The beeping sounds send signals or actually 'talk' to your phone company and its computerized switching system. That's how calls are made.

Now think of it. What if your phone was first connected to your own telephone computer. And what if in your own telephone computer you had a real genie that actually took your command and performed electronic magic on your phone lines. Far fetched? Read on.

WHAT KIND OF MAGIC

What if the system proved to be the fastest and most positive way to reach another person at another phone regardless of whether the phone is busy or whether that person is even near a phone. Enter Busy Buttons.

Busy Buttons is a miniature computer in a small black box. The box is nothing much to look at, measures only 1½" x 5" x 5¾" and in fact most people would probably hide it. The box plugs into the back of any telephone in your house or any multi-line telephone in your office. That's right, just plug it in. No installation, no wires to connect. Just plug it in.

HERE IT COMES

Now here comes the fairytale part. In that black box is indeed a real genie–a small creature so smart that it will understand every command you give it from your telephone's push buttons. No foolin'.

If you dial a number and that number is busy, you tell the genie you're upset by pressing the 'frustration' button–that's the button with a star on it. The genie will first redial that same number ten consecutive times the first minute and then once every two minutes thereafter until it reaches your party. When the call does go through, your genie will then signal you to pick up the phone. And you can still make calls and receive calls in between those times your genie is trying to reach the other party.

But what if there's no answer? After you let your phone ring for awhile, press the 'disappointment' button. The genie will then dial your number every ten minutes for up to ten hours and then signal you when somebody answers.

GENIES NEVER FORGET

"But what if the genie forgets the number it was dialing?" you might ask. The answer is quite logical. Genies never forget. In fact, you can own a genie so smart it will remember up to 176 numbers each up to 32 digits so you can not only dial long distance, you can use your genie to dial the entire 23 digit MCI or Sprint numbers in seconds. Your genie will recognize the tones, the pauses and faithfully dial your number accurately each time saving you tons of money on long distance charges.

"Too complicated," you might say. For your genie it might be but not for you. Remember, you use your own push button phone. There's no other attachment other than that dumb black box where your genie lives. And when you want to dial a number, you dial PAUL to reach Paul, MOM to reach your mom or HAIR to call your hair stylist. Remembering names is easy, remembering numbers is genie work.

THE REAL SHOCK

Ready for a real shock? You only need one genie to cover every telephone in your house or office. That's right. Unlike other auto dialers, one genie is all you need to turn every phone into this fully automatic system. But wait, there's more.

Genies talk differently. The American genie talks very rapidly in tones like most push button phones. There is even a Japanese genie that talks slowly and methodically in a pulsating sound similar to a rotary dial telephone. This means you can use Busy Buttons on push button or rotary dial telephones.

The Busy Button system is quite inexpensive. Genies you see have no minimum wage, are exempt from EEOC, EPA, OSHA, FDA and HEW regulations and don't mind putting in overtime or washing windows.

DIFFERENT VERSIONS

A 176 number Busy Buttons costs only **$200**–the 93 number version costs **$180**. If you want the Japanese Genie, you can have either model for $20 cheaper. And you can order Busy Buttons using your credit card by calling our toll-free number below. (Illinois residents add 6% sales tax.) Or send your check for the amounts listed above plus $4.00 for postage and handling to the address below.

When you receive your Busy Buttons computer just plug it in. That's right, plug it in. Then see how easy it is to program, how easy it is to redial a number either yourself or automatically. If you're not happy with the convenience of the Busy Buttons or the time and money you save from the day you install it, return it anytime within 30 days for a prompt and courteous refund including your $4.00 postage and handling charge. It won't cost you a penny and you won't insult the genie.

At the beginning of this advertisement we told you that the above story may be true or it may be false. Well it's true. There really is a genie in every Busy Buttons. And if you believe that, wait till you hear about our new computerized burglar alarm with its own built-in SWAT team. Order your Busy Buttons at no obligation, today.

JS&A® PRODUCTS THAT THINK

Dept. SN One JS&A Plaza
Northbrook, Ill. 60062 (312) 564-7000
Call TOLL-FREE 800 228-5000
In Nebraska Call 800 323-6400

Flag Use Tip

Dear Editor:

I wish to pass on a tip to readers of *SYNC* concerning flags. In games involving two players (and in other two-way situations), flags are often used to determine which group of data is to be used. The actual setting and resetting of flags would usually go something like this:

```
10 LET A=0
........
100 LET A=1 (or 100 LET A=A+1).
```

Suppose that you don't know if the flag is set, but you want to change its state. You have to type:

```
100 IF A=1 THEN GOTO 115
105 LET A=1
110 GOTO 120
115 LET A=0
```

A much easier way is to type

```
100 LET A=ABS(A-1).
```

This does the exact same thing with much less memory space.

Philip Gervais
714 5th Ave. So.
Clinton, IA 52732

ZX80/81 World

Dear Editor:

Wow! Am I ever impressed. I had no idea of the breadth of the ZX 80/81 world. As I have had my ZX 80 only two months, some things in the programming completely eluded me. My very first issue of *SYNC* (1:5) really opened up a lot of grey areas for me.

Anyhow, the main reason for this letter is the availability of the first 4 issues. If you have them on hand, please mail and bill. If not, do you know of an outlet for same.

I will appreciate any assistance.

Tony Wall

Ed.—At this point we have only a few copies of 1:2 and 1:4. Back issue orders must include payment of $2.50 per magazine. There are no other outlets.

The Great Circle Route

Dear Editor:

I enjoyed Chuck Dawson's "The Great Circle Route" (*SYNC* 1:5) and found it a good exercise to use it in conjunction with a standard mileage chart which gives distances in nautical miles. However, I felt that the addition of city names and approximate cost would make an interesting improvement. The cost figures are based on the average cost of Transcontinental U.S. flights and can be changed by adjusting the .009 figure in line 205.

A report code 5 (FULL SCREEN) will appear after destination LATITUDE prompt and it will be necessary to use CONTINUE.

The program changes are:

```
20 PRINT "ENTER DEPARTURE CITY"
21 INPUT A$
23 PRINT A$
100 PRINT "DESTINATION CITY"
101 INPUT B$
103 PRINT B$
185 PRINT A$;,#B$
205 PRINT "COST#$";.09*DIST;"####"
```

H.S. Wake
4171 Stettler Way
San Diego, CA 92122 ■

Make the most of your Sinclair Computer . . .

Software on Cassette!

MULTIFILE — Data Storage System An amazingly versatile multi-purpose filing system for the 16K ZX81. The program is menu-driven, and number, size and headings of files are user-definable. Both string and numerical files are catered for. Files may be created, modified, replaced, and searched, and are protected by an ingenious foolproof security system. Output to the ZX printer is also provided.

The program comes on cassette, together with three quality data cassettes for file storage, and comprehensive documentation, describing a host of applications for both business and personal use. If your ZX81 is bored with playing games, then this program will give it plenty to think about! . .$29.95 ($39.95 in Canada)

ZXAS MACHINE CODE ASSEMBLER Bored with BASIC? POKEING not your scene? Learn and program in machine code the easy way with this powfrul Z80 assembler, commissioned specially for the ZX81 & ZX80.

Standard Z80 nemonics are simply written into REM statements within your BASIC program. The assembly listings, together with addresses and assembled codes are displayed on the screen when assembled. The assembled code is executed with the USR function. The program uses 5K of memory and is protected from overwriting. Full documentation, including examples, is supplied with the cassette. This program is a must for all serious ZX81 & ZX80 users. . . .
$9.95 ($12.95 in Canada)

Last Minute Addition: ZXDB

The perfect complement to ZXAS assembler, ZXDB is a complete combined machine code disassembler and debugging program. May be used in conjunction with ZXAS and will leave about 9K of memory for your own program. Additional features include Single Step, Block, Search, Transfer and Fill, Hex Loader, Register Display and more. Executed by single keyboard entry. The combination of ZXAS/ZXDB plus one of our books will teach you all you need to know to program in machine codes.
ZXDB . . $9.95 ($12.95 in Canada)

Exciting Book Titles!

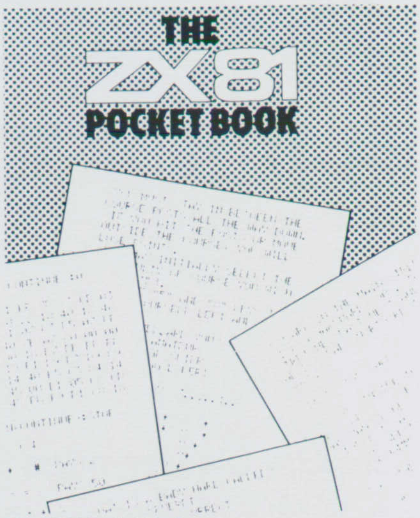
MACHINE LANGUAGE MADE SIMPLE FOR ZX80 and ZX81. A complete beginners quide to machine language programming. Go beyond BASIC and open new computer horizons! Finally find out what PEEK and POKE is all about. Machine language program enables more computing power in less space, faster running programs. The 120 pages of this book are packed with programming techniques, hints and tips; useful BASIC program to edit machine language; numerous sample routines; easy-to-use reference tables.
$19.95 ($23.95 in Canada)

UNDERSTANDING YOUR SINCLAIR ROM. A more advanced publication explaining the various ROM features.$19.95 ($23.95 in Canada)

ZX CHESS!

(for ZX81 and 8K/ZX80 both with 16K RAM)

A challenging chess programme, written in machine language, designed to operate in the ZX81 fast mode. ZX Chess allows you to select from 6 levels of play, choose either black or white, and enables castling and en passant moves. Unique "self-running" feature: you start the tape and when the chess board appears on the screen, start your game
ZX CHESS! Melbourne House. **$24.95** (29.95 in Canada)



The ZX81 Pocket Book
Written in the informative and clear style of the earlier, highly successful ZX80 Pocket Book, but with all new content. This is the ideal follow-up to the Sinclair manual, with application to both ZX81 and 8K ROM ZX80! The ZX81 Pocket Book begins with an exceptional 1K RAM programme (Pinning the Tail on the Donkey), which is followed by revealing chapters on String-Functions and Efficient Programming. Throughout there is a balance between serious computing concepts and fun programs. A particular emphasis is placed on the use of subroutines. Ohter chapters provide Hints 'n' Tips, Decimal Justification, Using Machine Code, Numeric Conversion, and ZX81 Adventure. Programs for both 1K and 16K machines include: Ski Run, Ball & Bucket, Etch-a-Sketch, Digital Clock, Standard Deviation, Dice Simulation, City of Alzan (a long adventure program), plus many others. The book contains 5 appendices containing ZX80 and ZX81 conversions, ZX81 module selector listing, solutions to problems in the book, ZX81 Basic command summary, and error code summary. The emphasis throughout is on a programming style designed to conserve memory, and demonstrate practical techniques to make your programs function better. Every Sinclair owner should have a copy right alongside his manual!
The ZX81 Pocket Book. by Trevor Toms, Phipps Associates.
136 pages. Spiral bound. $11.95 ($14.95 in Canada)

NOT ONLY 30 PROGRAMS FOR THE SINCLAIR ZX81 . . . BUT ALSO . . . detailed explanations and much much more. All programs designed to fit into the 1K memory of the ZX81. Includes such favorites as Star Wars, Lunar Lander, Blackjack, Mini Adventure. Also explanations of how programs were written, hints on how to write your own exciting programs, space-saving techniques, peeks and pokes and other "complicated" functions. . . . .
$14.95 ($16.95 in Canada)

Mail Orders to: **GLADSTONE ELECTRONICS** 901 Fuhrmann Blvd., Buffalo, NY 14203. (in Canada, mail to Gladstone Electronics, 1736 Avenue Rd., Toronto, Ont M5M 3Y7)

Name

Address

City State Zip

Charge to ☐ Visa ☐ MasterCard

Card No.

Expiry

☐ Check ☐ Money order (Sorry, no CODs)

Amount enclosed

Full replacement warranty all tapes.

Please send

Quantity	Description	Price each
	CASSETTES	
	Multifile -Data Storage System	$29.95 ($39.95 Cdn)
	ZXAS - Assembler	$9.95 ($12.95 Cdn)
	ZXDB - Disassembler/debugger	$9.95 ($12.95 Cdn)
	ZX CHESS!	$24.95 ($29.95 Cdn)
	BOOKS	
	Machine Language Made Simple for ZX80 & ZX81	$19.95 ($23.95 Cdn)
	Understanding Sinclair ROM	$19.95 ($23.95 Cdn)
	The ZX81 Pocket Book	$11.95 ($14.95 Cdn)
	...Not Only 30 Programs for Sinclair ZX81...	$14.95 ($16.95 Cdn)
	Shipping charge, all orders	$1.50

FREE! Write for our new catalogue of software and hardware add-ons.

sync notes

Paul Grosjean

MicroAce Discontinues U.S. Operations

MicroAce has discontinued its U.S. operations as of mid-December 1981, but arrangements have been made for repair services on MicroAce equipment. MicroAce owners with problems should contact:

Bob Ward
3176 Oak Knoll
Los Alamitos, CA 90720

Problems and complaints may be also directed to MicroAce's U.K. address:

MicroAce Compshop
14 Station Road
New Barnet
Hertsfordshire EN5 1QW
United Kingdom

MicroAce is also looking for distributors who will represent MicroAce in the U.S. for continued sales of the video upgrade board. If you are interested, write MicroAce at the U.K. address above.

Sinclair to Replace Defective MicroAce 8K ROMs

Sinclair has announced that defective MicroAce 8K ROMs will be replaced by Sinclair if the ROM is sent along with $10 to:

Sinclair Research Ltd.
50 Staniford St.
Boston, MA 02114

Sinclair Policy Change on Technical Phone Inquiries

Sinclair has discontinued the policy of dealing with technical questions on the Sinclair computers by phone. Among the reasons cited for this change are: 1) the volume of sales; 2) the increasing proportion of kit sales which raise more complex questions difficult to answer by phone; 3) many inquiries have to do with applications rather than the actual computer operation.

However, the repair policy remains unchanged. If you are having problems with your computer, it can be returned to Sinclair for either repair or replacement free of charge within 90 days of purchase.

An attempt will be made to answer technical questions for owners of the Sinclair computers addressed to: Sinclair Research Ltd., 4 Sinclair Plaza, Nashua, NH 03061.

SYNC NOTES U.K. Win a £1,000

[*Ed.—We received the following announcement from Prestel*:]

British Telecom's Prestel—the world viewdata servie—is offering a prize of £1,000 to the designer of the best Prestel adaptor for the Sinclair ZX81.

Telesoftware—computer programs distributed from a central source via teletext or viewdata to computers in schools, homes and offices—is a growth area on Prestel. Following the initiative of the Department of Industry in funding the Council for Educational Technology's Educational Telesoftware Project, many program publishers are becoming active with program libraries of up to 1,000 pages strong being established.

Since the Sinclair ZX81 is yet without Prestel adaptation, Prestel is offering a prize of £1,000 which will be awarded to the designer of the ZX81 adaptor which combines best the elements of low price, elegant design and practical robustness.

The closing date for the competition is March 14, 1982. A working prototype capable of being modified so as to receive approval for attachment to the telephone network must be submitted. Designs will remain the property of the designer.

Further details, specification, and entry forms are obtainable from Tony Sweet, Prestel Headquarters, Telephone House, Temple Avenue, London EC4Y OHL. Telephone 01-583 9811.

The object the ZX81 competition is to design a system which will be capable of loading ZX81 software contained on a Prestel frame into the RAM of the machine and be ready to run.

Rules:

a) The adaptor must work to Prestal frames in CET format.

b) It must be capable of production on a commercial basis, i.e., using readily available components.

c) It should be "in the spirit of the ZX81," i.e., low price, practical robustness, and efficient design.

d) It must be capable of modification to receive approval for the attachment to the telephone network.

e) A detailed circuit diagram/parts must be submitted with the completed entry. This will be treated in strictest confidence by BT.

f) Designs and prototypes will remain the property of the designer.

g) The judges decision will be final. No correspondence will be entered into over the acceptance/non-acceptance of entries.

h) Entries must be received by 14 March 1982 and must be demonstrated by the designer before acceptance. Arrangements will be made for designers to have editing access to Prestel as necessary.

i) Completed entry forms must be submitted to BT before designs are submitted.

SYNC Program Listings

Readers should note the following conventions used in the program listings in this issue:

or • = Used in PRINT statements to show necessary spaces.

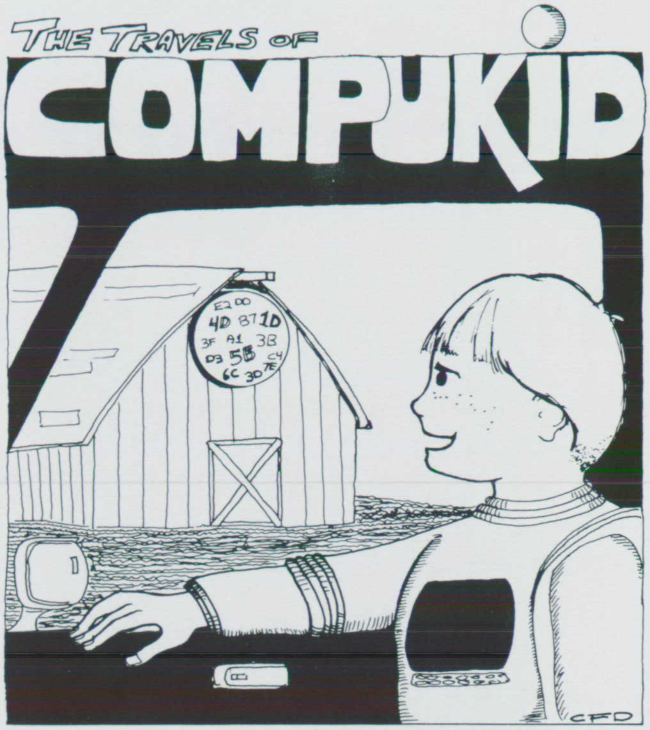
"A" (shift) = Used in PRINT statements to indicate graphics; in this case use the graphic on shift A.

INPUT = Used in PRINT statements to show that the keyboard key or token should be used instead of spelling out the word.

Sinclair 8K ROM Problems?

In SYNC Notes (*SYNC* 1:5) we gave a test to check whether you had received a defective ROM. The following letter from Nigel Searle will clarify this matter further:

> Dear Customer:
>
> The recent mention in Sync magazine of a possible bug in our 8K BASIC ROM could be misleading.
>
> The correct value of 2^{32} is 4,294,967,296. This is rounded to 8 significant digits and displayed as 4,294,967,300. 2^{32-1} is 4,294,967,295 which, when rounded to 8 significant digits, is also displayed as 4,294,967,300.
>
> This is the appropriate result for a computer of finite (8 digit) capacity.
>
> The bug referred to in Sync will give 1,288,490,200 when 1 is subtracted from 2^{32} (4,294,967,300). Only if your ROM has this problem should you return it to us for replacement.
>
> Sincerely,
>
> NH Searle
>
> Nigel H. Searle



"Now there's a real hex sign!"

ARTIC COMPUTING

NEW

MOVING AHEAD WITH ZX SOFTWARE

16K RAM PACKS ONLY £35.00

ZX CHESS & ADVENTURES

PROGRAMS FOR THE ZX81/80 INCLUDING –

ZX CHESS I reduced to: **£8.00**

ZX CHESS II now only: **£14.00**

ADVENTURES

Very popular machine code program, with six levels of play and an analysis option. Unbeaten except by:

A new improved version, with a faster response time, seven levels of play, and in addition a recommended move option.

Exciting machine code games with instant response, choose from the range below.

ADVENTURE 'A', £5.00 — You find yourself stranded on an alien planet. Can you reach your ship and escape?

ADVENTURE 'B', £7.00 — In a jungle clearing you come across an Inca temple. You must break in, collect treasure and escape alive. Beware. Includes a cassette save routine.

ADVENTURE 'C', £7.00 — You are unfortunate enough to be drawn to an alien cruiser. Can you reach the control room and free yourself or will they get you first? Includes a cassette save routine.

1K GAMES PACK £3.00

and many more. For a catalogue giving full details, please send a S.A.E. to.

GALAXY WARRIOR £3.00

Fast and exciting interactive animated graphics game. Hunt clingons and go through black holes....

ARTIC COMPUTING
396 JAMES RECKITT AVENUE
HULL, HU8 OJA. U.K.

KOPAK GIVES YOU THE POWER!

KOPAK'S™ Touch-A-Matic gives you the power to type more accurately and much faster. No wires, no soldering, simple to install. Our unique vinyl overlay guides your fingers to the correct keys. Touch typing now possible with your Sinclair* or MicroAce*!

$9.95

The KOPAK™ Sinclair* female Connector gives you the power to use the Z-80 bus. Edgecard connectors specifically made for Sinclair* computers. • 46 Pins, 23/23 • All pins are gold plated • polarizing pin for correct alignment everytime! Available with Solder-Tail, Wire-Rap and PCBoard.

$14.95

'KOPAK has the POWER to deliver!'

Order by Phone or Mail,
Charge it with MasterCharge or Visa

KOPAK Creations Inc.
Dept. SY
448 West 55th Street
New York, NY 10019
(212) 757-8698

KOPAK™ is now offering The Source* 'America's Information Utility'. The Source* gives you the power to access a wide array of services including: • Barter • Shopping • Business Information • Career Network • Electronic Mail • Medical Advice • UPI News Service and much more. A modem is required to use The Source*.

Source ID $100.

More POWER on the way! KOPAK'S™ Coming Creations:

The KOPAK™ RS-232 Interface
The KOPAK™ Bus
The KOPAMATIC™ Keyboard
The KOPAK™ Printer
KOPAK™ Memory Products
The KOPAK™ Disc

© 1982 KOPAK Creations Inc.
TM Trademark of KOPAK Creations Inc.
Sinclair* is a trademark of Sinclair* Research LTD.
MicroAce* is a trademark of MicroAce*

perceptions

David B. Ornstein

4K and 8K ROM Command Conversions

In the last issue of *SYNC* (1:6) I began a discussion of the conversion of programs from the 4K Integer Basic to the 8K F.P. (Floating Point) Basic and covered *expressions* and *functions*. In this issue I will conclude the discussion by detailing the processes necessary for the conversion of *commands* from one ROM to the other.

The following commands are 100% transferable from a program on one ROM to a program on the other ROM:

LET
NEW
RUN
CONT (INUE)
STOP
GOTO
RAND (OMISE)
GOSUB
REM
IF....THEN
INPUT
PRINT
LIST
POKE
CLEAR
RETURN

Because a parallel for each of the commands in the 4K ROM exists in the 8K ROM, I will list alphabetically all the 8K ROM commands. When possible, a 4K ROM conversion will be given. (The names of the 4K commands will be given in brackets.)

CLS
[**CLS**]

The CLS command is used to clear the entire (TV) screen. On a 4K ROM system or an 8K ROM system with less than 3.25K of RAM, the CLS command works identically (i.e., it creates a minimal display file). On an 8K machine with more than 3.25K of RAM, a full display file, padded with spaces, is created. The command is generally interchangeable between ROMs.

COPY

The COPY command is used to print the screen, as is, on the ZX Printer. As the printer cannot be used with a 4K ROM system, there is no simulation available to perform a parallel function.

DIM
[**DIM**]

DIM is used to create arrays and matrices. On the 8K ROM, both numeric (F.P.) and string (character) arrays are available. On the 4K ROM, only numeric (Integral) arrays are available. As noted earlier, (see Perceptions, *SYNC* 1:6), there is no way to use floating point numerics on a 4K system. String arrays are also, generally, out of the question.

The 4K ROM's integral arrays are useful in that they save 3-bytes per number. If you have an application that really needs integral arrays because of memory size constraints, a method can be derived to emulate them. As usual, the gain in space will result in a slow-down of the system.

Imagine a situation in which you want a 100-element intergral array. Running the 4K Basic, to allocate it, you might type:

```
DIM A(100)
```

This would instruct the system to reserve 200 bytes for the elements of the array (2-bytes each). To reference the 40th element, you might use the statement:

```
LET J = A(40)
```

My 8K emulation is relatively simple. In place of the original DIM statement, use the following:

```
DIM A$(200)
```

Then, to reference the 40th element, you would use an expression like:

```
LET J = CODE A$ (39*2+1)+ CODE A$(39*2+2)*256
```

This method accesses the appropriate element's value by reading its 2 data bytes and combining them, putting the result into the variable J.

To change the value of an element, you might use the following sequence:

```
10 LET TEMP=INT(J/256)
20 LET A$(39*2+1 TO 39*2+2) = CHR$ (J-TEMP*256)+CHR$(TEMP)
```

This would set the 40th element to the value of J and is equivalent to:

```
LET A(40)=J
```

Generally, the subexpression to use if you want to reference the Xth element is:

```
CODE A$ ((X-1)*2+1)+ CODE A$((X-1)*2+2)*256
```

FAST

FAST is a command used to control the video mode of the ZX81. The ZX80 (i.e., 4K ROM) is *always* in FAST mode.

FOR...NEXT (STEP)
[**FOR...NEXT**]

The FOR and NEXT commands are used to set up a loop in the program. The 8K version has two features not implemented on the 4K version. The first is that you may have non-integral values for the looping variable. This cannot be simulated on the 4K ROM. The other feature, a STEP modifier, can be simulated.

If you run the following program:

```
10 FOR J=1 TO 10
20 PRINT J
30 NEXT J
```

you will see the numbers from 1 to 10,

stepping by 1, printed on the screen. If, on an 8K ROM, you run the program:

```
10 FOR J=1 TO 10 STEP 2
20 PRINT J
30 NEXT J
```

you will see the numbers from 1 to 10, *stepping by 2*, printed on the screen.

The first program is equivalent to the following:

```
10 LET J=1
20 PRINT J
30 LET J=J+1
40 IF J<11 THEN  GOTO 20
```

Notice that in line 30, the value stored in J is incremented by one. To make this run as per the 8K program, add line 5 as follows:

```
5 LET ST=2
```

and change line 30 to say:

```
30 LET J=J+ST
```

Now it will run and print the numbers from 1 to 10 on the screen, stepping by *two*.

LLIST

The LLIST command is used to list out the program on the ZX Printer. As is the case with the COPY command, it is not possible to use the printer, and, thus, the printer commands, on the 4K ROM.

LOAD
[**LOAD**]

The LOAD command is used to bring a previously SAVEd program back into the computer. The 8K version uses a file name (possibly null) to specify which program on the tape you want to LOAD. The 4K version simply takes the next program on the tape and LOADs it. The formats on 4K and 8K programs involve such a differential as to make it a fruitless venture to attempt to LOAD a program SAVEd on a 4K machine into an 8K machine, or vice versa.

LPRINT

The LPRINT, used to print a line out to the ZX Printer, has the same constraints applied to it as does LLIST.

PAUSE

The PAUSE command is used to delay for a given number of frames on the TV (16ms), while continuously scanning the picture. Barring a moderately complex machine language program, it cannot be simulated. (For those interested, I suggest my article "The ZX80 Keyboard" (*SYNC* 1:2), which describes the keyboard/display scanning routine, as a good starting point.)

PLOT/UNPLOT

These commands are used to PLOT particular graphics points on the screen. There is no completely parallel function on the 4K ROM. It must be done with a PRINT statement.

SAVE
[**SAVE**]

The SAVE command, used to SAVE the current program on tape for later retrieval, is available on both ROMs. The only difference is that the 8K version takes a name as an argument.

SCROLL

SCROLL (SCreen ROLL) is used to roll the entire screen up one line. Although no parallel exists in the 4K Basic, it can be (and has been) simulated with machine language. See Ian Logan's article "Screen Scrolling" (*SYNC* 1:4) for more details.

SLOW

The SLOW command, like the FAST command, is used to select the video mode of the ZX81. It is non-functional on a ZX80 with an 8K ROM and cannot be simulated on a 4K machine.

16K RAM Pack Schematic Corrections

A number of readers have noted some problems with the 16K RAM pack schematic printed in *SYNC* 1:5. The following corrections should be made:

1) On IC 7 no pin 13 is listed, but it should be shown and connected to pin 6.

2) On ICs 3, 4, 5, 6 pin 15 is not shown, but it should be shown and connected to ground.

3) On IC 4 a line is shown going from pin 7 to an undesignated point on IC 8. On IC 8 this point should be labeled as pin 6.

kitchen sync

Alan Groupe, Michael Tardiff, and Ivan Zatkovich

To Build or not to Build? A Reveiw of the Sinclair ZX81 Kit

As has already been said in the pages of *SYNC* and elsewhere, Clive Sinclair deserves some sort of medal for introducing a much improved version of an already popular computer and selling the new version at a lower price. You don't find deals like that often these days, on either side of the Atlantic.

After you decide that the ZX81 is the computer for you, your only remaining decision is whether to save $50 and buy the kit version, or whether to go the fast and easy route and buy a "plug-in-and-run" ZX81.

Should you build your own ZX81? If you've never built anything electronic before, no. If you haven't used a soldering iron (a small, 25-watt or less, narrow-tipped iron with rosin-core solder) before, no. If your patience is not one of the things for which your friends admire you, no.

But if you've built kits before, Heathkit or otherwise, if you can solder with confidence, and if you have a couple of spare hours on hand—yes! You can have a good time, save enough money to make it all worthwhile, experience the warm inner glow that comes from using something you assembled yourself, and be reasonably sure that your ZX81 will work the first time you turn it on.

The engineering of the computer is topnotch, as is the packaging. The board is silk-screened and solder-masked, and poses no problem if you have a soldering iron with a small tip and some fine-gauge solder. The instructions for assembly, however, leave a bit to be desired if you like (or need) to have things spelled out step-by-step for you.

Getting It Together

Our kit came carefully packed in a plastic foam container, not just thrown in a box. The components were packed in little bags although they weren't grouped in any order that paralleled the assembly instructions. We were told by a friend, however, that his kit was packaged in a large cardboard box rather than plastic foam like ours. (We'll get a third opinion when our other two kits arrive—there are three of us, and one ZX81 can't satisfy all of us at once...)

The assembly instructions aren't Heathkit-style "step-by-step"; they might as well read, "Put the components in the proper locations and solder them in." If you read the instructions carefully and work slowly, this won't be an impediment. But, if you can't wait, and rush ahead, making assumptions along the way, you'll blow it, like we did. Since the board uses plated-through holes (the inside walls of the holes are metal-plated just like the traces), it can be frustratingly difficult to remove misplaced components after you've soldered them in.

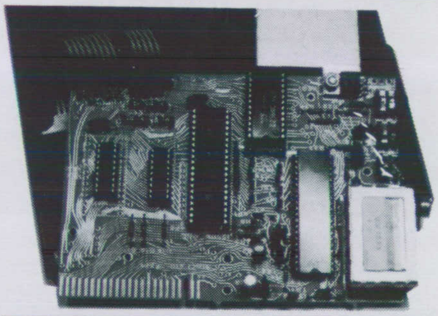
The instruction sheet starts off by recommending that you read all of the instructions carefully. A good idea, but "memorize carefully" is what it should read. The arrangement of the assembly instructions and the "warnings and hints" is such that you must remember the "warning and hints" as you go through the assembly instructions. If the "warnings and hints" had been mixed in with the assembly instructions, we would have made three fewer mistakes than the three mistakes we made. We didn't have anyone to caution us, though; now that we've told you, you shouldn't have any trouble.

There are many extra "unused" holes on the board (it is common on plated-through boards to use holes to connect traces from one side of the board to the other), and the board is so small and tightly packed that it is easy to place a lead in the wrong hole if you are not careful. A picture of a completed board showing proper component layout would have been a great help. We've included such a photograph with this article. (Figure 1).

The parts list includes along with the component name (R3, etc.) and the component value a description of the marking on the components. This is handy for people who may have not memorized the EIA resistor color code, and in identifying some oddly-marked parts (like some of the capacitors that are marked in nanofarads (1000 pf), an uncommon unit in the United States).

Our kit had the correct number of each part, but our friend's kit was missing one memory chip and a resistor, and had two extra transistors and couple of spare resistors of different values. Be sure to check your parts against the parts list before you start building—Sinclair will gladly replace shorted parts, and it's disappointing to have to stop assembly in the middle to the kit to wait for a part to come in the mail.

Figure 1.



The instructions suggest that the resistors are pre-cut and bent, a time-saver to the kit builder. However, both the 470 ohm and most of the 1K ohm resistors come full-length. This, as we discovered ten solder connections too late, is no accident. The pre-cut resistors are installed flat on the board, while the full-length ones are mounted on-end, sort of standing up. While a warning to this effect is indeed contained in the instructions, this warning is not where we were looking when we installed them; hence, we put many of these resistors in wrong and had to make a trip to Radio Shack. Also note that some parts (like

Make your "LITTLE" ZX81 *work like a* BIG *computer with* BYTE-BACK *modules*

INSTANT INFORMATION WITH BYTE-BACK'S MD-1 MODEM

In Stock! *only $99.95*

Use your phone to connect your "LITTLE" ZX81 or ZX80 to the "LARGEST" computer networks in the world. With **BYTE-BACK's MD-1 MODEM** connected all you do is dial a phone number (usually local), press a few keys and watch the data appear on your TV screen.

You can have immediate access to:

- **DOW JONES:** Current stock transactions and statistics on financial performances, plus complete investment and business information.
- **UPI:** The United Press International Database allows you to read virtually any information from UPI wires, including news, weather and sports plus newspaper columns (often available days before they appear in your local newspaper).
- **MEDICAL ADVICE:** Home Health-Care Database allows you to diagnose common health-related problems. It's very valuable help during medical emergencies.
- Review thousands of items from *The Wall Street Journal, Barrows* and other major newspapers.
- **PLUS:** Movie reviews, discount shopping, barter-trade network, games, weather, jobs available nationwide, etc.

There's no minimum monthly charge for all of this. You just pay for the time you are actually connected. Rates start at less than 5 per minute. As with phone rates they are much cheaper in the evenings. Complete details included.

RS-232 PORT INCLUDED

As an extra bonus an RS-232 port is provided to allow you to drive all standard RS-232 peripherals.

BYTE-BACK'S MD-1 Modem with your ZX80/1
makes all of this possible!

16K MEMORY MODULE (M-16)

only ***$69.95*** *NEW in stock!*

BYTE-BACK'S 16K memory module plugs right into the back of your ZX81 (or ZX80, with or without 8K Basic).

But unlike other 16K memory modules, up to three **BYTE-BACK M-16** memory modules can be connected at one time to get a total of 48K.

BYTE-BACK'S BB-1 CONTROL MODULE

$59.00 In Stock!

- **8 Independent Relays**
  (with LED status indicators)
- **8 Independent TTL Inputs**
  (with Schmitt trigger buffers)

The BB-1 **CONTROL MODULE** plugs directly into the **ZX80/1** expansion port. It accepts inputs from remote switches, thermostats, photocells, A/D's etc. Your **ZX80/1** can read the status of all 8 inputs by the use of a single PEEK command.

By using a single POKE command you can change and latch the status of each of the 8 relays to control lights, motors, pumps, alarms, recorders, water heaters, furnaces, solar heating systems, model railroads, 8 bit parallel port printers and it can even do remote data logging. You hams can even use it as an automatic keyer to send code.

A comprehensive manual is included that has complete application details.

By using the **BB-1** with your **ZX80/1** to control lights or other energy saving devices you can get a TAX CREDIT based on the cost of your **ZX80/1** and your **BB-1 CONTROL MODULE.** Details included.

☐ BB-1 Kit and Manual	$59	£30
☐ BB-1 Wired and Tested and Manual	$69	£35
☐ BB-1 Blank PC Board and Manual	$29	£15
☐ Modem Kit	$99.95	£49

☐ M-16 Kit	$69.95	£35
☐ M-16 Wired and Tested	$79.95	£39
☐ M-16 Blank PC Board	$19.95	£10
☐ Modem Wired and Tested	$119.95	£59
☐ RS-232 Port (W & T); not shown	$59.95	£30

Shipping and Handling $4.95

ORDER PHONE (803) 532-5812

Bill My ☐ VISA ☐ Mastercharge

Exp. Date ______________________ Card No. ______________________

Name ______________________

Address ______________________

City/State/Zip ______________________

Mail to:

BYTE-BACK Co. • Rt. 3, Box 147 • Brodie Rd. • Leesville, S.C. 29070

- VISA
- MASTER CHARGE
- CHECKS

ORDERS MAILED
First Class (U.S.A.)
Air Mail (England)

REMEMBER: With* BYTE BACK *modules you are NOT limited to using only one module at a time!

R30, a 10-ohm resistor) are used in the U.S. version of the kit, but not the U.K. Be sure you've read the instructions carefully to decide which parts are or aren't used. Figure 2 will help you in proper component placement.

Also included are two "resistor packs." These packs have a right end and a wrong end, and must be installed the right way. The instructions state that the common end is marked by a white dot—ours had no dots, white or otherwise, but instead a black square towards the middle but off to one side. No big problem, really—the end closer to this black square is the common end. If the instructions had reminded us, "Install the resistor packs, remembering to place the marked end near the "C" silk-screened on the circuit board," we probably wouldn't have soldered them (both!) in backwards. Desoldering a seven-pin part from a plated-though board, we discovered, is very close to impossible without destroying the part or the board. We had to wait a week for the replacements to come from Sinclair to pay for our error.

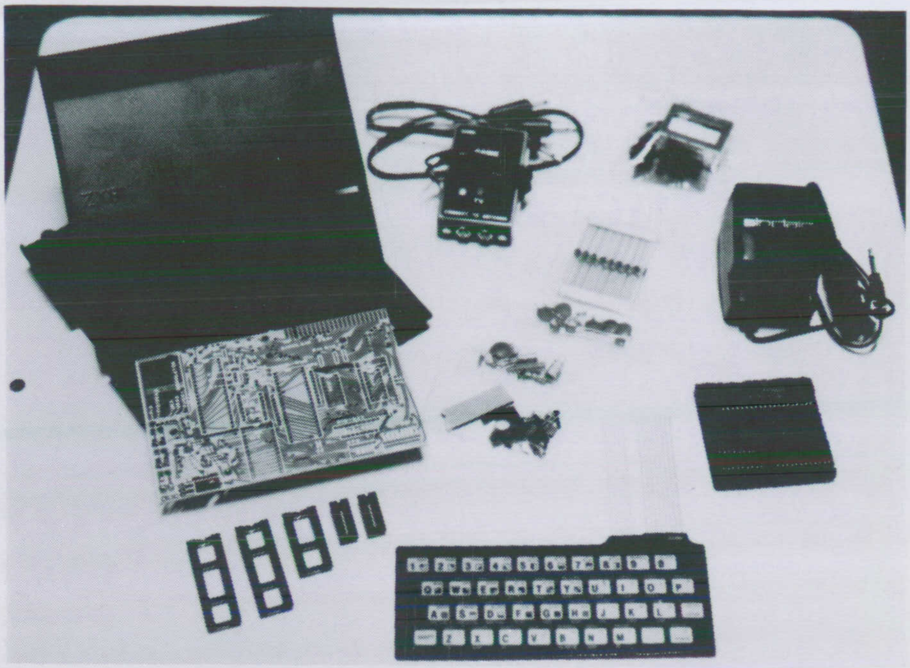
We discovered a spare part at the end of our assembly, which we soon decided was the ceramic filter (it looks like a three-legged capacitor). It had been left out of the "install all of these parts" list. The kit won't work without the filter, so you should (as we did) put it in, even though you're not explicitly told to do so.

The installation of the modulator posed a problem we couldn't solve ourselves. According to the instructions, the two leads go in the holes marked Fr/UK1 and UK2. Common sense told us that this may be wrong, since we knew that neither "Fr" nor "UK" have television systems exactly like "US." We, of course, wanted to use the holes marked "USA," but there are three such holes and only two leads. We then decided to look at the schematic—wrong move. Apparently there are several versions of the ZX81 kit, and the schematic appears to be an interesting combination of all of them. (Printing one schematic and one instruction set for the world is probably one of the ways Clive manages to keep prices so low...)

Thoroughly confused, we decided to call Sinclair for assistance. They informed us that there are two variations of the kit being delivered to the US—a UHF kit with a two-lead modulator using the Fr/UK1 and UK2 holes, and a VHF kit with a three-lead modulator that uses the "USA" holes. Problem solved. According to our friend, later kits like his are packaged with a supplementary instruction sheet that makes all this clear.

The keyboard for the ZX81 is a self-contained unit, unlike the ZX80's, which used the bottom of the printed circuit board and a stick-on overlay. A thin film with conductors "printed" on it serves to connect the keyboard unit to the computer; this film slips into a connector you install on the printed circuit board. It is a good idea to clean the ends of the keyboard connector film before you plug it into its connector. Light rubbing with a pencil eraser will ensure a good connection. Slipping the keyboard straps into the connector proved to be a little tricky, but an even pressure and some verbal urging does the trick.

Figure 2.



We checked the voltage produced by the power supply before we plugged the ICs into their sockets, just in case something else had gone wrong in the assembly: we couldn't wait another week for replacement ICs to come in the mail. If you have access to a voltmeter, you can check the power supply by plugging the AC adapter into the ZX81's power jack, then measuring the voltage across pins 2 and 3 (middle and right-hand) of the voltage regulator (which is attached to the metal heat sink). If you find five volts there, everything's ok so far. If you don't happen to have a volt meter, try using the "burnt thumb" method—apply power to the board and place your thumb on the plastic case of the regulator. If your thumb starts smoking, the regulator is using more current than the designers planned, and you probably have a short somewhere. Check your soldering, and look for solder bridges across traces.

Be sure that you unplug the kit before installing the ICs, unless you want to zap a couple chips.

The Moment of Truth

After all our troubles, the machine worked the first time we plugged it in. We put the case together (it's held together by screws instead of these funny plastic things used on the ZX80), plugged it in, and entered our first program. After playing around with our new toy for a while, we became ZX81 converts (and thus our two more kits on the way). Our only remaining question was why Sinclair put the vents on the bottom of the case. Must have been so the heat could fall out.

From start to finish the whole building process took about five hours, including a little backtracking. Considering the $50 savings offered by the kit version, you earn about $10 an hour, have a good time (barring disaster), and end up with a great computer for the lowest price around.

All things considered, the quality of the kit is excellent; Sinclair didn't cut corners anywhere. Everything Sinclair has ever made has been ingenious, well-built, and surprisingly inexpensive, and the ZX81 is not exception. Sinclair remains hard to beat for engineering excellence—but tonight, in Benton Harbor, Heathkit sleeps easy.

[Authors' P.S.: Since writing the review, we have received both a UHF and a VHF kit from Sinclair. There are some significant differences between the two which will be covered in a future column.] ■

Writing For SYNC

Paul Grosjean

If you have material you want us to consider for publication, we are very much interested in looking at it. If it fits our editorial needs at that time, we will send you a "Transfer of Copyright Agreement" to sign and payment for your article. On the average we pay about $20 per printed page in *SYNC*. When you submit material, we ask that you keep the following in mind:

1) Type your manuscript on standard typing paper (one side only) with at least one inch margins all around.

2) Use the double space setting for your text throughout.

3) If you want your manuscript returned, enclose a self-addressed stamped (do not use a postage meter) envelope. If you want to be sure that we have received your work, enclose a self-addressed postcard.

4) Be sure to put your name, address, and phone number on the top of the first page in one corner. In the other corner put the machine requirements of your article or program (for example, 8K ROM; lK RAM. 8K ROM; l6K RAM. 4K ROM; lK RAM). Remember that our readers have a variety of ROMs and RAMs and they are not happy to find out after they have entered a program that it does not fit their machine. Put the title or a short form of the title on each page in the upper left corner. Paginate on the upper right corner.

5) Type with normal use of capital and lower case letters. Do not type everything in capitals in your text (in programs, however, do use capitals since that is what your computer uses). This applies also to headings and subheadings. Show subheadings by leaving extra space above and centering.

6) Underlining means that those letters should be italics when printed. So underline only when you mean "Use italics here."

7) Paragraphs must be indented (5-8 spaces is usual). *Do not* use extra lines to show paragraphs.

If your article includes programs or listings, please keep these items in mind:

1) We prefer camera ready copy of programs and listings whenever possible because this substantially reduces the risk of typographical errors. Carbon ribbon typewriters make excellent copy. Printers and regular typewriters which give a sharp, clear image usually can be used, but make sure that the ribbon is dark. Of course, when the ZX Printer becomes available in the U.S., printouts from it will be acceptable. In addition, we would like to have the programs submitted on cassette with several saves, especially if the program is over 1K.

2) Type the program so that it will look just like the screen display including all spaces that are necessary or that the computer puts in automatically. Do not use extra spaces where the computer does not use them. This is a big help to the reader in checking whether he is entering the program correctly and helps him reduce copying mistakes.

3) Show necessary spaces in PRINT statements with a # mark.

4) Program notes which help the reader to understand what is going on are helpful. These may be given as side notes on the same line as the program line or at the end of the program with line numbers for matching. In either case keep the width of the notes the same as the program listing.

5) If you use graphics, be sure to specify in the notes which key to use to get the graphic.

6) If possible, make suggestions for adapting your program to fit other machine requirements. For example, if you have a program that takes 2K RAM, tell the reader where to shorten it to squeeze it into lK if possible. If you are writing for an 8K ROM, supply the changes necessary for the 4K ROM if possible. Readers like to know where they can make changes in the programs to vary the results. Point these out also.

7) Type your program single spaced.

8) Be sure to indicate in your article how to RUN the program and what the reader should expect to see on the screen when he has done so.

9) Follow the emerging conventions for ZX80/81 programs: a) number program lines by 10's unless you have a reason to do otherwise; b) avoid using letters that can be confused with numbers and vice versa; c) use consecutive designations for strings and variables; d) identify your program with a REM statement.

10) Provide the SYNCSUM (see the Perceptions column in this issue). Other checksums may be used if there is a good reason to do so, but the process for finding them should be explained.

Photos, illustrations, charts, and diagrams usually add to an article. Again, we prefer copy that we can use directly without redrawing. Illustrations can be larger than the expected final form because we can reduce them, but they should not be smaller. All charts, diagrams, listings, illustrations, photos, tables, and programs should be labeled such as Figure 1, Listing 1, or Table 1 and referred to in the text in that way rather than as "the table below" or "the following lines" because we may not be able to do it that way in our layout. It is even helpful to put all the figures, tables, etc. at the end of your article.

If you can supply your text and program listings on disk, include the information on the type of disk system you have. We would prefer that form if it is compatible with our equipment.

Following these suggestions will help us a great deal in using your material.

8K ROM

try this

This columns will feature short programs to show off your computer, impress your family and friends, and tickle your imagination when *SYNC* arrives at your place. We invite your contributions. Address them to: Try This, SYNC, 39 E. Hanover Ave., Morris Plains, NJ 07950.

With your computer in SLOW mode, type in:

```
10 REM YNC and hit NEWLINE
```

Type in:

```
POKE 16513,56 and hit NEWLINE.
POKE 16517,147 and hit NEWLINE.
```

Then hit NEWLINE again.

With your computer in FAST mode, type in:

```
10 PRINT CHR$ (INT (RND*8)+2;
20 RUN
```

After observing the results, try it again in SLOW mode (hit SLOW and NEWLINE and then type RUN and NEWLINE).

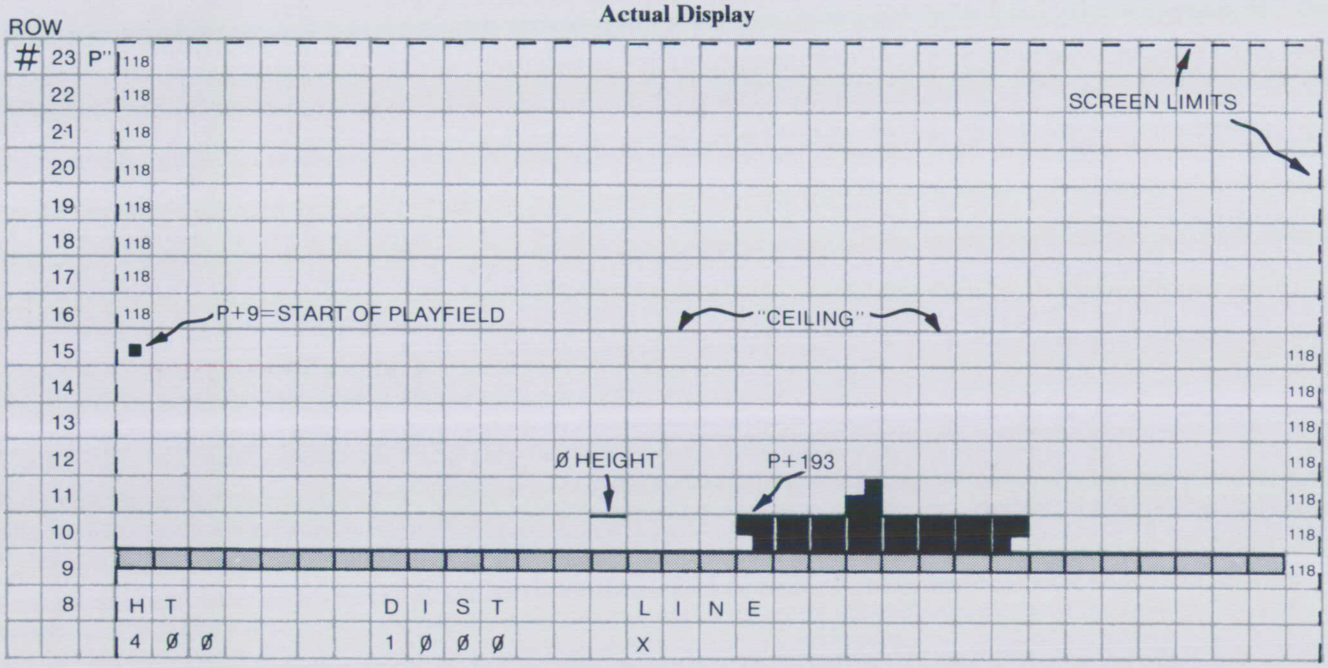
James Grosjean, 50 Kings Road, Chatham, NJ 07928.

4K ROM
1K RAM

How to Invent a Game— Inside Flattop Lander (MCD)

Jon Bobst

Figure 1.
Actual Display



"How to make a million dollars and NEVER pay taxes," by Steven Martin. "First, get a million dollars..."

As computerists, the million-dollar question that you and I had after learning ZX80 fundamentals was, "Okay, now what do I do with it?"

First...Get an Idea

The general idea for *Flattop Lander* was a natural progression from my other "aircraft" games. Good ideas are all around you—take a look.

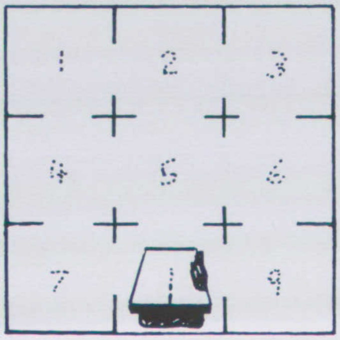
Probably everyone has seen a movie showing landings on aircraft carriers, whether of WWII propcraft or of modern-Navy jets. There are two images that we retain: the view from the aircraft (pilot's view) and the scene from deck or from another ship. Your first decision after getting an idea is to choose a display image.

For a descent pilot's view sequence with its separate "frames" for a three dimensional effect, several *hundred* addresses in RAM must be used to store the different characters needed. With only 1K of RAM available for both display *and* basic program, the pilot's view option is out.

The next best thing is to combine a 2-D display (side-view) of the landing with a routine that requires the player to "get into" the display. In other words, require an input that *involves* the player, making him/her construct a mental image or "mind's eye" picture as in Figures 1 and 2.

This "window" is recognizable as the standard 3 x 3 input matrix, but with a 3-D twist: 5 means straight ahead and 9 means

Figure 2.
Mental "Window"



Jonathan Bobst, ZETA Software, P.O. Box 3522, Greenville, SC 29608-3522.

down to the right, etc. The Basic display of Height and Distance are not really necessary, because the player can see his position on screen. But the LINE is *not* optional—that tells the pilot which way to turn: left "<", right ">", or "X" for on-line for a straight landing. (That is part of the deck flagman's job.)

So, in the planning stage for your display, refine your idea to a scene that allows both room for movement *and* room for the Basic statements and any Basic printouts you may need.

Display Construction

First, you must estimate just how *much* display you can have—enough to make the "play" visually interesting, but not so much that the program will crash or stop on error code 5 (no more room on screen).

Remember that each space assigned to a screen display means one less address that can be used in the Basic program (and vice versa).

In *Flattop Lander*, since the last 400 feet in height are the most important to a pilot, the actual "play field" can be set up as 5 rows of 32 spaces for aircraft movement, 1 row for the zero-height of the flight deck and carrier characters, and 1 row of sea surface. Thus, 7 rows of 32 spaces plus 7 "118's" as end-line markers plus 8 "118's" to move the play field to mid-screen totals 239 addresses—leaving 700+ for the Basic program.

Second, work out how you would construct your display in Basic. For *Flattop Lander*, it would take around 20 statement lines *and* 2 or 3 seconds in execution time. The obvious advantage of an MCD version is that, while it may take more "know-how" to write, execution time is typically *less* than half a second...short enough for *anyone's* attention span.

Next, translate your Basic display into Machine Code decimal values.

Z80 Source Coding for Flattop Lander (4K/1K)

This subroutine produces *instantly* a blank field and 1 row of seasurface characters. The aircraft and carrier will be POKEd in later via Basic, enabling movement experiments of the aircraft and/or carrier. The *idea* is to produce 239 spaces on screen *quickly* so that by POKEing only 9 characters in different places, the whole display will be accomplished in the minimum of time. See Figure 5.

Figure 5.

Decimal Code | **Z80 Source Statement** | **Comment**

Step 1: Initialize the subroutine with the address held in "D-FILE"

```
42                  LD HL,(nn)        ;put display address
12                  n=12              ;held at 16396 (D-FILE)
64   =16396         n=64              ;into HL register-pair
35                  INC HL            ;point to next screen address
```

Step 2: Loop eight times for a column of "118's"

```
6                   LD B,n            ;put the following number
8                   n=8               ;into the b register/counter
62                  LD A,n            ;put the following number
117                 n=117             ;into the A register
60                  INC A             ;add 1 for "118"
119                 LD(HL),A          ;put number in A into the
                                      ;address in HL
35                  INC HL            ;point to next address
16                  DJNZ e            ;subtract 1 from B and if not
249                 (e=255-249)       ;zero, jump back 6 values (to 62)
```

This roundabout way of loading "118" into a screen address is necessary because "118" in Z80 means "HALT" processing, while in ZX80, it means "end of line." Either way, you cannot use it in a 1 REM subroutine.

Step 3: Loop six times for 6 rows of 32 spaces (blank play field).

```
6                   LD B,n            ;B is row counter
6                   n=6
197                 PUSH BC           ;put the number in B into the
6                   LD B,n            ;"save" stack
32                  n=32              ;B is now free to count spaces
54                  LD(HL),n          ;put chr code into screen address
0                   n=0               ;chr code is for a "space"
35                  INC HL            ;point to next screen address
16                  DJNZ e            ;decrement B and Jump Non-Zero
251                 e=4               ;minus 4 values (to 54)
62
117
60                  (same as above) ;same as above but for 1-"118"
119
35
193                 POP BC            ;put number saved on the stack
16                  DJNZ 3            ;into the B register, subtract 1
240                 e=15              ;and jump back if not 0 (to 197)
```

Step 4: PRINT 1 row of sea surface, CHR$(11)

```
6                   LD B,n            ;load B with 32
32                  n=32
54                  LD(HL),n          ;load screen address with chr "n"
11                  n=11
35                  INC HL            ;point to next address
16                  DJNZ e            ;subtract, test, and jump not-0
251                 e=4               ;backwards 4 values (to 54)
62
117
60                  (same as above) ;for end of line "118"
119
35                                    ;point to next address available
                                      ;after MCD
```

Step 5: Reset system variable addresses and exit subroutine.

```
34                  LD (nn),HL        ;load the last address in HL
14   =16398         n=14              ;into both "DF-EA"
64   (DF-EA)        n=64
34                  LD (nn),HL        ;and "DF-END"
16   =16400         n=16
64   (DF-END)       n=64
62                  LD A, n           ;load A with the row number
8                   n=8               ;after MCD (23-15=8 for F.L.)
50                  LD (nn),A         ;put that row number into 16421
37                  n=37              ;meaning #8 is the next available
                                       row
64   =16421         n=64
201                 RET               ;return to Basic program
```

Entering and Testing an MCD

Count the number of values in the subroutine (55 for *Flattop Lander*) and enter that many "boxes" into line 1:

1 REM □□□ etc. :□ = shifted A

You can use any character, or even 1, 2, 3, etc. The idea is to *reserve* some non-changeable addresses so they can hold MC values. The first box's address is 16427. (16424=21, 16425=0 16426=254 or "REM".)

Line 2 in *Flattop Lander* holds 8 boxes in REM for storing carrier characters. Otherwise, use 2 REM B as a buffer against accidently moving the cursor to line 1: that's a big "no-no" after you enter the MC values!

Okay, you have 1 REM with a line of boxes and 2 REM with either more boxes for character storage or "B" for buffer. The next step is to scroll line 1 off-screen. "How?" you may ask. "I only have the 4K chip."

You do not *need* the 8K ROM chip to scroll one of only two lines off screen, and you do not have to enter a lot of "dummy PRINTs" as bulk, either.

Here is the trick—with the cursor *below* line 1, enter this command on the edit line:

POKE 16403,2

and ZAP!, line 2 is now at the top and line 1 is off-screen. (Look on page 122 of your ZX80 manual for the reason it works. Hint: 16404 always contains 0 when the line number is less than 256.)

Now enter the input routine to POKE the 1 REM boxes with MC values. See Figure 3.

If you write down your MC values in a 4-column table (read left to right) and then "INPUT" from it, one 4-value row at a time, it is easy to spot entry errors.

GOTO 100 and enter the MC values, then SAVE on tape in case of entry or coding errors. Lastly, on the edit line:

LET Q=USR(16427)

and BINGO! your initial display springs *instantly* to life from a lot of no-longer-mysterious numbers.

Well, maybe it doesn't...in which case, unplug your "crashed" ZX80, plug it back in to reset the ROM from hardware, LOAD the partial program, and edit out lines 110 and 120. GOTO 100 for a complete printout of the MC code table and look for entry error. If none, recheck the decimal values of your subroutine. (My usual mistake is to miscalculate the offset values for DJNZ and other "Jump Relatives.")

Now that you have an instant MCD, you have to POKE some characters into it. If you use 2 REM for character storage, you can use the same input routine by changing line 100. Add 5 to the address of the last box in 1 REM for the address of the first box in 2 REM. *Flattop Lander* has eight characters in 2 REM, so line 100 is FOR X=16486 TO 16493.

Finishing an MCD with POKEd Characters

For reasons of scale and lack of RAM, only one character is used for the aircraft in *Flattop Lander*: 27, or a period. 20, or a "-", looked better airborne, but not landed. The more characters you have moving around the screen, the more complicated and lengthy your Basic POKE-routine will be.

How the game's display is tied together is shown in Figure 4.

Before and after the above statements are game routines for entering values from the input matrix and changing the value of "C"; one line to assign a random value to "L" as "wind current"; lines to calculate Height, Distance and which way to turn to stay on the approach Line; a Score, Crash, or new-approach input routine; and an end-game/exit routine, all in Basic. That sounds like a lot of processing between displays, but "flicker" time *remains* under half a second! See the advantage of an MCD, now?

For your own games, you can invent or modify displays and rules to suit yourself. In this way you can see one of the best reasons for buying a microcomputer rather than a "game machine." With a computer, you are not restricted by someone *else's* viewpoint of what is FUN.

In Part 2, we will see how to develop "game" routines in Basic that will make your program do what you want it to do. ■

Figure 3.

```
100 FOR X=16427 TO (16427+(XX-1))   ;XX=the number of MC values
110 INPUT Y                          ;FLATTOP LANDER = "TO 16481"
120 POKE X,Y                         ;this loads the box w/value
130 PRINT PEEK (X),                  ;for a 4-column "running" display
140 NEXT X                           ;of POKED address-values
```

Figure 4.

```
 40 LET C=9                         ;9 added to "P" = 1st field addr.
 90 LET Q=USR(16427)                ;call subroutine in 1 REM
100 LET P=PEEK(16396)+PEEK          ;"P" = 0th address of screen
    (16397)*256                      display
110 POKE P+163,132                  ;132 is "bridge" chr of carrier
120 LET Y=16486                     ;Y = first address in 2 REM
130 FOR X=P+193 TO P+200            ;start carrier at 193rd space
140 POKE X,PEEK(Y)                  ;"print" chr held in 2 REM
150 LET Y=Y+1                       ;point to next chr address
160 NEXT X
170 POKE P+C,27                     ;aircraft position and chr
180 PRINT "HT","DIST","LINE"        ;1st of 3 Basic print lines
```

The Game of Life Revisited— An Assembly Version

Richard Booth

The *Game of Life*, printed in *SYNC* 1:2 (pp. 28-30), was written in Basic. This Basic program may be replaced with an assembly subroutine which displays each succeeding generation within a fraction of the time needed by the Basic program. Each generation is constructed and then displayed by using another assembly subroutine for creating a display file based on "A Display File in Machine Code" by Dr. I. S. Logan in the same issue (pp. 13-15).

Program to Enter Assembly Subroutines

Figure 1 is a listing of the program for entering the Hex formatted assembly instructions. Line 100 contains the space which will be occupied by these instructions. As always, once the subroutines have been entered, *do not LIST the program* because the system will hang up.

The assembly routine is entered one byte at a time. A backspace character, "/", is provided in case an error is made in entry. This backspace can be used more than once. When the subroutine is completely entered, type in "END" to stop.

Assembly Subroutines

Figure 2 is a listing of the display file subroutine adapted from Dr. Logan's article. When the routine is RUN, the display will show a palette of 10x30 locations surrounded by a border.

Richard Booth, 12875 Highland Rd., Highland, MD 20777.

Figure 1: Program to Enter Hex Format.

```
100 REM  [247 shift D's]
200 LET V=-1
210 LET V=V+1
220 IF (V/50)*50=V THEN CLS
230 IF (V/10)*10=V THEN PRINT
240 INPUT H$
250 IF H$="" THEN GO TO 240
260 PRINT H$;"#";
270 IF H$="END" THEN STOP
280 IF H$="/" THEN GO TO 310
290 POKE 16427+V,16*CODE(H$)+CO
    DE(TL$(H$))-476
300 GO TO 210
310 LET V=V-1
320 GO TO 240
```

Figure 3 is a listing of the *Game of Life* subroutine. Its structure, similar to that of the Basic *Game of Life*, is shown in flowchart form in Figure 4. The rules for the birth and death of cells are the same as in the Basic version.

Entering the Program

After entering the program in Figure 1, type in the Hex Format column in the Display File subroutine in Figure 2. Type 'END" when this has been done. To check to see if this has been correctly entered, type LET A USR(16427). The palette should appear immediately. If it does not, check the program.

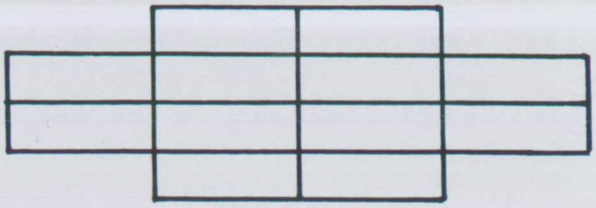
Figure 2: Display File Assembly Subroutine.

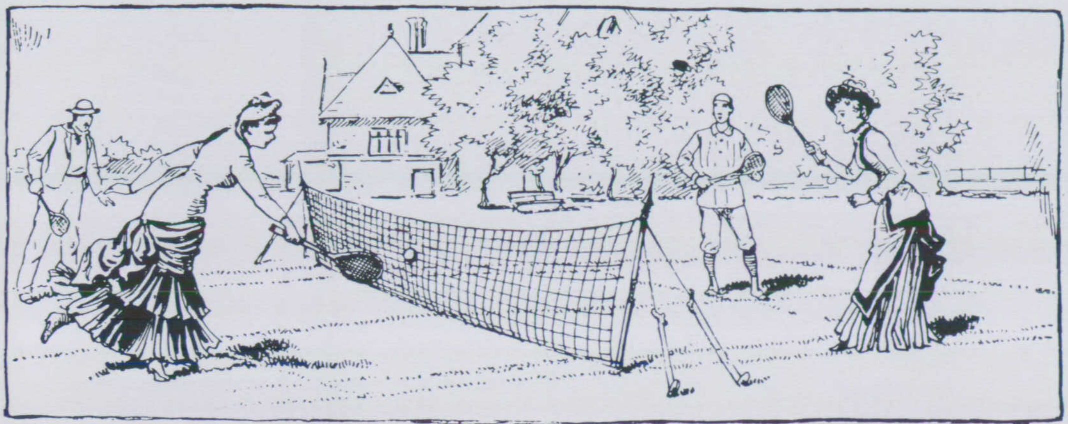
Decimal Address	Hex Address	Label	Mnemonic Format		Hex Format
16427	402B	START	LD	HL,(D-FILE)	2A 0C 40
	E		INC	HL	23
	F		CALL	EDGER	CD 57 40
	4032		LD	B,10	06 0A
	4	LINE	PUSH	BC	C5
	5		LD	(HL),9	36 09
	7		INC	HL	23
	8		LD	B,30	06 1E
	A	SPACE	LD	(HL),0	36 00
	C		INC	HL	23
	D		DJNZ	SPACE	10 FB
	F		LD	(HL),9	36 09
	4041		INC	HL	23
	2		CALL	DLIM	CD 5E 40
	5		POP	BC	C1
	6		DJNZ	LINE	10 EB
	8		CALL	EDGER	CD 57 40
	B	END	LD	(DF-EA),HL	22 0E 40
	E		LD	(DF-END),HL	22 10 40
	4051		LD	A,11	3E 0B
	3		LD	(LINE-CTR),A	32 25 40
	6		RET		C9
	7	EDGER	LD	B,32	06 20
	9	EDGE	LD	(HL),9	36 09
	B		INC	HL	23
	C		DJNZ	EDGE	10 FB
	E	DLIM	LD	A,117	3E 75
	4060		INC	A	3C
	1		LD	(HL),A	77
	2		INC	HL	23
	3		RET		C9

puzzles & problems

A Rectangular Problem

Let's start off with a simple (?) counting problem. As in all puzzles of this nature you only get one chance so study the diagram carefully. Your problem is to ascertain how many rectangles are depicted in the illustration at the right. Are there 12? Are there 20? You have five minutes to decide.



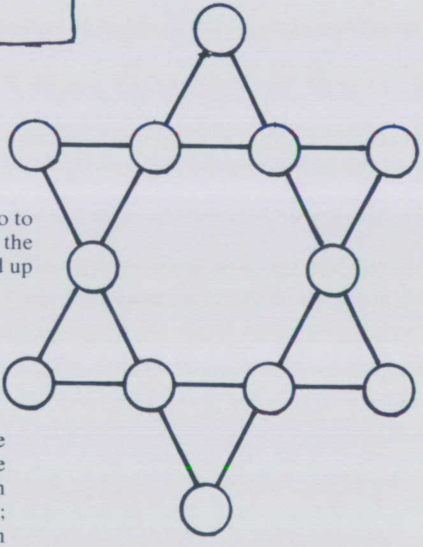


Who's Who?

Three couples are out playing tennis at the Shady Knoll country club. The names of the players are Fred, Tom, Joe, Mary, Nancy and Patty. Joe and his sister are sitting out the first game while Fred and Tom's wife are pictured here playing against Mary and Nancy's husband. From your vantage point at the side can you tell who is married to whom?

The "26" Puzzle

Can you place the numbers 1 through 12 in the 12 circles that go to make up the star pictured at the right in such a manner that the values in the four circles, in each of the six rows or circles, add up to 26?



A Word Square Problem

GEORGEN
RERENTS
LESSTEN
RATRIOT
RANLENT
TENTRAN
RENTGAS

Good word-square puzzles are few and far between. Below we have pictured seven seven-lettered words. The letters in each word have been jumbled about. Your problem is to sort out each word and then rearrange the words top-to-bottom so as to form a "word-square"; that is, the words in each row will also be spelled out in seven columns top-to-bottom.

The Flock of Geese

Two friends, passing a woman with a flock of geese, made a wager as to who should guess nearest at their number, without actually counting, one maintaining that there were not more than thirty, the other that there were over forty. On asking the market-woman which was right, she replied, "If I had as many more, and one-half as many more, and one-fourth as many more, I should have one short of a hundred. Now puzzle it out for yourselves." What was the number of the flock? (This puzzle is from that great old Victorian puzzle book *Puzzles Old and New* by Professor Hoffman)

That's it for this issue folks. Remember, if you have any puzzles that you would like to share with the readers of *SYNC* send them in, and, if Merlin uses them, he will send you a copy of one of his famous *Merlin's Puzzler* books.

Your Editor, Charles Barry Townsend

Charles Barry Townsend

Answers on page 33.

Figure 3: Game of Life Assembly Subroutine.

Starting address 16498 decimal.

```
Hex      Hex                        Mnemonic
Address  Format       Label         Format              Comment
4072:    DD 21 23 00  LIFE:         LD    IX,0023H
4076:    ED 4B 0C 40                LD    BC, (400CH)   ;Display File Address
407A:    DD 09                      ADD   IX,BC
407C:    DD E5                      PUSH IX             ;1st PALETTE LOCATION
407E:    0E 0A                      LD C,0AH            ;10 Rows
4080:    06 1E        COUNT:        LD B,1EH            ;30 Rows
4082:    16 00        LOOPX:        LD D,00H
4084:    DD 7E FF                   LD A,(IX-01H)       ;Begin Search for Neighbors
4087:    CD F9 40                   CALL TEST           ;Call TEST routine
408A:    DD 7E 01                   LD A,(IX+01H)
408D:    CD F9 40                   CALL TEST           ;Call TEST routine
4090:    DD 7E E0                   LD A,(IX-20H)       ;Call TEST routine
4093:    CD F9 40                   CALL TEST           ;Call TEST routine
4096:    DD 7E DF                   LD A,(IX-21H)
4099:    CD F9 40                   CALL TEST           ;Call TEST routine
409C:    DD 7E DE                   LD A,(IX-22H)
409F:    CD F9 40                   CALL TEST           ;Call TEST routine
40A2:    DD 7E 20                   LD A,(IX-20H)
40A5:    CD F9 40                   CALL TEST           ;Call TEST Routine
40A8:    DD 7E 21                   LD A,(IX-21H)
40B7:    82                         ADD D
40B8:    DD 77 00                   LD (IX+00H),A
40BB:    DD 23                      INC IX
40BD:    10 C3                      DJNZ LOOPX          ;to LOOPX for next Row
40BF:    0D                         DEC C               ;Last Row?
40C0:    28 08                      JR Z,A              ;Finished Search, STart
                                                         SCORing
40C2:    DD 23                      INC IX STSCOR
40C4:    DD 23                      INC IX
40C6:    DD 23                      INC IX
40C8:    18 B6                      JR COUNT
40CA:    DD E1        STSCOR:       POP IX              ;1st PALLETTE LOCATION
40CC:    0E 0A                      LD C,0AH            ;10 Rows
40CE:    06 1E        BGNROW:       LD B,1EH            ;30 Columns
40D0:    DD 7E 00     SCORE:        LD A,(IX+00H)       ;Memory Location
40D3:    FE 03                      CP 03H              ;No cell, 3 neighbors
40D5:    28 0E                      JR Z,CELL           ;To CELL
40D7:    FE 82                      CP 82H              ;Cell, 2 neighbors
40D9:    28 0A                      JR Z,CELL           ;To CELL
40DB:    FE 83                      CP 83H              ;Cell, 3 neighbors
40DD:    28 06                      FR Z,CELL           ;To CELL
40DF:    DD 36                      LD (IX+00),00       ;No Cell
40E3:    18 04                      JR CELL+1           ;Skip over CELL
40E5:    DD 36        CELL:         LD (IX+00),00       ;Create Living Cell
40E9:    DD 23                      INC IX              ;Next Location
40EB:    10 E3                      DJNZ SCORE          ;To SCORE if not end of row
40ED:    0D                         DEC C               ;Row counter
40EE:    28 08                      JR Z,SREND          ;Last Row?
40F0:    DD 23                      INC IX
40F2:    DD 23                      INC IX
40F4:    DD 23                      INC IX
40F6:    18 D6                      JR BGNROW           ;Next Row
40F8:    C9           SREND:        RET                 ;Back to Basic
40F9:    FE 5F        TEST:         CP 5FH              ;Living Cell?
40FB:    F8                         RET M               ;Return to Counting if no Cell
40FC:    14                         INC D               ;Increment counter
40FD:    C9                         RET                 ;Return to Counting
402B:    2A 0C 40     START:        LD HL,(400CH)       ;Display File address
402E:    23                         INC HL
402F:    CD 57 40                   CALL EDGER          ;Call Edge Routine
4032:    06 0A                      LD B,0AH
4034:    C5           LINE:         PUSH BC
4035:    36 09                      LD (HL),09H
4037:    23                         INC HL
4038:    06 1E                      LD B,1EH
403A:    36 00        SPACE:        LD (HL),00
403C:    23                         INC HL
403D:    10 FB                      DJNZ SPACE          ;Loop to SPACE
403F:    36 09                      LD (HL),09H
4041:    23                         INC HL
4042:    CD 5E 40                   CALL DLIM           ;Row Delimiter
4045:    C1                         POP BC
4046:    10 EB                      DJNZ LINE           ;Line
4048:    CD 57 40                   CALL EDGER          ;Edge Routine
404B:    22 0E 40     END:          LD (400EH),HL       ;(DF-EA)
404E:    22 10 40                   LD (4010H),HL       ;(DF-END)
4051:    3E 0B                      LD A,0BH
4053:    32 25 40                   LD (4025),A         ;Line Counter
4056:    C9                         RET
4057:    06 20        EDGER:        LD B,20H
4059:    36 09        EDGE:         LD (HL),09H
405B:    23                         INC HL
405C:    10 FB                      DJNZ EDGE           ;Loop to EDGE
405E:    3E 75        DLIM:         LD A,75H            ;Row Delimiter
4060:    3C                         INC A
4061:    77                         LD (HL),A
4062:    23                         INC HL
4063:    C9                         RET
```

Next, in line 290 change 16427 to 16498. Run the program again and enter the Hex Format column in the *Game of Life* subroutine in Figure 3.

The Basic Portion of the *Game of Life* program is listed in Figure 5. The first part of this program is also from Dr. Logan's article. Replace lines 200-320 of the program entered from Figure 1 which are already in memory by entering lines 200-320 in Figure 5.

If possible, save this result. Errors may be corrected either by POKEing or by starting again.

Larger Field for Larger Memories

For readers with additional memory a larger area can be created. For example, a 20x30 field can be created by:

1) Load the program.
2) Change line 240 to read "...A>600 ..."
3) POKE 1635,20
   POKE 16511,20
   POKE 16589,20
4) SAVE the result.

Figure 5: Game of Life—Basic Portion.

```
100 REM, [Assembly Programs]
200 LET A=USR(16427)
210 PRINT "PRESS NO. OR 0"
220 INPUT A
230 IF A=0 THEN GO TO 280
240 IF A<1 OR>A 300 THEN GO TO
    220
250 LET A=PEEK(16396)+PEEK(1639
    7)*256+34+A+((A-1)/30)*3
260 POKE A,-128*(PEEK(A)=0)
270 GO TO 220
280 PRINT "PRESS NEWLINE"
285 FOR I=1 TO 100
290 LET A=USR(16498)
300 INPUT A$
305 IF A$="S" THEN STOP
310 NEXT I
320 STOP
```

Running the Program

After "RUN" has been typed, the display palette appears. Enter a starting generation by typing in position numbers as in Dr. Logan's article. When the starting configuration is complete, type "0" NEWLINE, to display the second generation. Each succeeding generation is displayed by typing NEWLINE. To stop the program, type "S".

A random starting generation may be entered by using the lines in Figure 6 as replacements in Figure 5.

Figure 6: Line Replacements for Random Start.

```
210 FOR I=1 TO 100
220 LET A=RND(300)
230
270 NEXT I
```

The $\$149^{95}$ personal computer.

Introducing the Sinclair ZX81

If you're ever going to buy a personal computer, now is the time to do it.

The new Sinclair ZX81 is the most powerful, yet easy-to-use computer ever offered for anywhere near the price: only $149.95* completely assembled.

Don't let the price fool you. The ZX81 has just about everything you could ask for in a personal computer.

A breakthrough in personal computers

The ZX81 is a major advance over the original Sinclair ZX80—the world's largest selling personal computer and the first for under $200.

In fact, the ZX81's new 8K Extended BASIC offers features found only on computers costing two or three times as much.

Just look at what you get:

- Continuous display, including moving graphics
- Multi-dimensional string and numerical arrays
- Mathematical and scientific functions accurate to 8 decimal places
- Unique one-touch entry of key words like PRINT, RUN and LIST
- Automatic syntax error detection and easy editing
- Randomize function useful for both games and serious applications
- Built-in interface for ZX Printer
- 1K of memory expandable to 16K

The ZX81 is also very convenient to use. It hooks up to any television set to produce a clear 32-column by 24-line display. And you can use a regular cassette recorder to store and recall programs by name.

If you already own a ZX80

The 8K Extended BASIC chip used in the ZX81 is available as a plug-in replacement for your ZX80 for only $39.95, plus shipping and handling—complete with new keyboard overlay and the ZX81 manual.

So in just a few minutes, with no special skills or tools required, you can upgrade your ZX80 to have all the powerful features of the ZX81. (You'll have everything except continuous display, but you can still use the PAUSE and SCROLL commands to get moving graphics.)

With the 8K BASIC chip, your ZX80 will also be equipped to use the ZX Printer and Sinclair software.

Order at no risk**

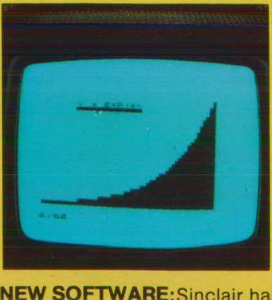
We'll give you 10 days to try out the ZX81. If you're not completely satisfied, just return it to Sinclair Research and we'll give you a full refund.

And if you have a problem with your ZX81, send it to Sinclair Research within 90 days and we'll repair or replace it at no charge.

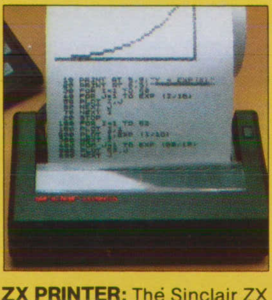
---

*Plus shipping and handling. Price includes connectors for TV and cassette, AC adaptor, and FREE manual.

**Does not apply to ZX81 kits.

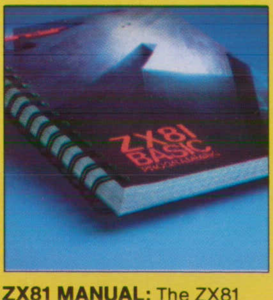


NEW SOFTWARE: Sinclair has published pre-recorded programs on cassettes for your ZX81, or ZX80 with 8K BASIC. We're constantly coming out with new programs, so we'll send you our latest software catalog with your computer.



ZX PRINTER: The Sinclair ZX Printer will work with your ZX81, or ZX80 with 8K BASIC. It will be available in the near future and will cost less than $100.

16K MEMORY MODULE: Like any powerful, full fledged computer, the ZX81 is expandable. Sinclair's 16K memory module plugs right onto the back of your ZX81 (or ZX80, with or without 8K BASIC). Cost is $99.95, plus shipping and handling.



ZX81 MANUAL: The ZX81 comes with a comprehensive 164-page programming guide and operating manual designed for both beginners and experienced computer users. A $10.95 value, it's yours free with the ZX81.

The $99^95 personal computer.

Introducing the ZX81 kit

If you really want to save money, and you enjoy building electronic kits, you can order the ZX81 in kit form for the incredible price of just $99.95.* It's the same, full-featured computer, only you put it together yourself. We'll send complete, easy-to-follow instructions on how you can assemble your ZX81 in just a few hours. All you have to supply is the soldering iron.

How to order

Sinclair Research is the world's largest manufacturer of personal computers.

The ZX81 represents the latest technology in microelectronics, and it picks up right where the ZX80 left off. Thousands are selling every week.

We urge you to place your order for the new ZX81 today. The sooner you order, the sooner you can start enjoying your own computer.

To order, simply call our toll free number, and use your MasterCard or VISA.

To order by mail, please use the coupon. And send your check or money order. We regret that we cannot accept purchase orders or C.O.D.'s.

CALL 800-543-3000. Ask for operator #509. In Ohio call 800-582-1364. In Canada call 513-729-4300. Ask for operator #509. Phones open 24 hours a day, 7 days a week. Have your MasterCard or VISA ready.

These numbers are for orders only. For information, you must write to Sinclair Research Ltd., 2 Sinclair Plaza, Nashua, NH 03061.

sinclair

AD CODE	PRICE†	QTY.	AMOUNT
ZX81	$149.95		
ZX81 Kit	99.95		
8K BASIC chip (for ZX80)	39.95		
16K Memory Module (for ZX81 or ZX80)	99.95		
Shipping and Handling	4.95		$4.95
		TOTAL	

MAIL TO: Sinclair Research Ltd., One Sinclair Plaza, Nashua, NH 03061.

NAME ____________________

ADDRESS ____________________

CITY/STATE/ZIP ____________________

† U.S. Dollars

01SY

Some Good Starting Generations

To begin your exploration of the *Game of Life*, try entering some of the following generations. The reason for my name choices should become evident.

"Migrating L": 50,79,109,110,111

"E-Lights": 135,136,137,138,139,165, 167,169

"Hawk": 74,77,104,105,106

"D-Hive": 100,101,130,132,160,162, 190,191

"Melting Snow": 100,101,102,103,104, 130,132,134,160,161,162,163,164

Watch especially the interaction between the different colonies and the borders. Different starting generations will create different succeeding patterns depending upon where they are placed within the palette.

For 8K ROM/16K RAM

This assembly version of the *Game of Life* may also be implemented on 8K ROM/16K RAM machines. Since there are quite a few changes, the entire revised listings are given below in Figure 7. The program to enter Hex format (Figure 1) may be used with the following line replacements:

220 IF INT(V/50)*50=V THEN CLS
230 IF INT(V/10)*10=V THEN PRINT
290 POKE 16514+V,16*CODE H$+CODE H$(2 TO 2)-476

Use this revised hex-loading program to enter the assembly subroutines listed in Figure 7. Then type in the Basic program listed in Figure 8. The program does not need to press NEWLINE for each succeeding generation to be displayed, since a pause is taken between each display.

A next-generation version of the *Game of Life* might make use of pixel graphics, and a non-flashing display. This would effectively quadruple the "world" size, and some very complex patterns could be observed. ▚

Figure 8. Game of Life

Basic Portion (8K ROM)

```
 100 REM  ASSEMBLY PROGRAM
 200 LET A=USR 16514
 210 PRINT "PRESS NO. OR 0"
 220 INPUT A
 230 IF A=0 THEN GOTO 280
 240 IF A<1 OR A>600 THEN GOTO 2
20
 250 LET A=PEEK (16396)+PEEK (16
397)*256+34+A+INT ((A-1)/30)*3
 260 POKE A,128*(PEEK (A)=0)
 270 GOTO 220
 280 FOR I=1 TO 1000
 290 LET A=USR 16562
 300 PAUSE 40
 310 NEXT I
 320 STOP
```

Figure 9. Line Replacements for Random Start (8K ROM)

```
210 FOR I=1 TO 100
220 LET A=RND*300+150
230      [DELETE]
270 NEXT I
```

Figure 7. Game of Life Assembly Routine (8K ROM; 16K RAM)

Decimal Address	Hex Address	Label	Mnemonic Format	Hex Format
16514	4082	START:	LD HL,(D_FILE)	2A 0C 40
	E		INC HL	23
	6		CALL EDGER	CD A3 40
	9		LD B,20	06 14
	B	LINE:	PUSH BC	C5
	C		LD (HL),8	36 08
	E		INC HL	23
	F		LD B,30	06 1E
	4091	SPACE:	LD (HL),0	36 00
	3		INC HL	23
	4		DJNZ SPACE	10 FB
	6		LD (HL),8	36 08
	8		INC HL	23
	9		CALL DLIM	CD AA 40
	C		POP BC	C1
	D		DJNZ LINE	10 EB
	F		CALL EDGER	CD A3 40

Figure 4: Flowchart for Game of Life Subroutine.

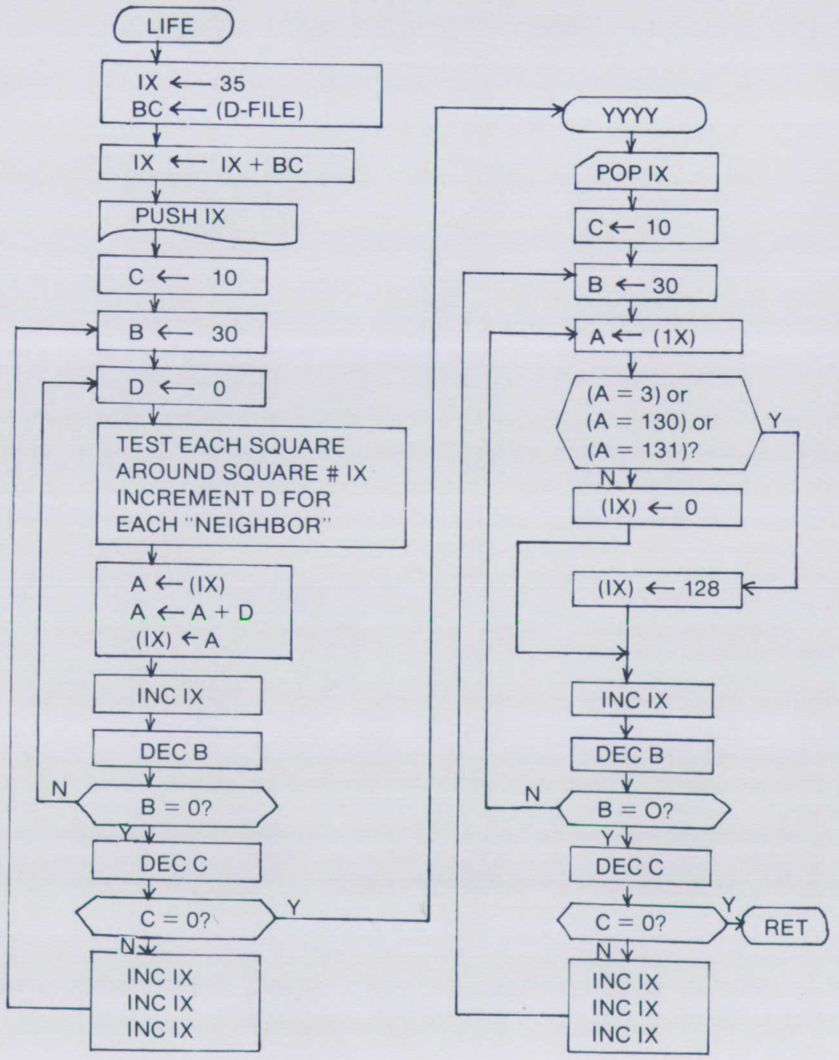


Figure 7 continued

```
Decimal  Hex      Label    Mnemonic              Hex
Address  Address           Format                Format
         40A2              RET                   C9
            3     EDGER:   LD B,32               06 20
            5     EDGE:    LD (HL),8             36 08
            7              INC HL                23
            8              DJNZ EDGE             10 FB
            A     DLIM:    LD A,117              3E 75
            C              INC A                 3C
            D              LD (HL),A             77
            E              INC HL                23
            F              RET                   C9
         40B0              NOP                   00
            1              NOP                   00
16562       2     LIFE:    LD IX,35              DD 21 23 00
            6              LD BC,(D_FILE)        ED 4B 0C 40
            A              ADD IX,BC             DD 09
            C              PUSH IX               DD E5
            E              LD C,20               0E 14
         40C0     ZZZZ:    LD B,30               06 1E
            2     XXXX:    LD D,0                16 00
            4              LD A,(IX-1)           DD 7E FF
            7              CALL TEST             CD 39 41
            A              LD A,(IX+1)           DD 7E 01
            D              CALL TEST             CD 39 41
         40D0              LD A,(IX-32)          DD 7E E0
            3              CALL TEST             CD 39 41
            6              LD A,(IX-33)          DD 7E DF
            9              CALL TEST             CD 39 41
            C              LD A,(IX-34)          DD 7E DE
            F              CALL TEST             CD 39 41
         40E2              LD A,(IX+32)          DD 7E 20
            5              CALL TEST             CD 39 41
            8              LD A,(IX+33)          DD 7E 21
            B              CALL TEST             CD 39 41
            E              LD A,(IX+34)          DD 7E 22
         40F1              CALL TEST             CD 39 41
            4              LD A,(IX)             DD 7E 00
            7              ADD D                 82
            8              LD (IX),A             DD 77 00
         40FB              INC IX                DD 23
            D              DJNZ XXXX             10 C3
            F              DEC C                 0D
         4100              JR Z,YYYY             28 08
            2              INC IX                DD 23
            4              INC IX                DD 23
            6              INC IX                DD 23
            8              JR ZZZZ               18 B6
            A     YYYY:    POP IX                DD E1
            C              LD C,20               0E 14
            E     TTTT:    LD B,30               06 1E
         4110     VVVV:    LD A,(IX)             DD 7E 00
            3              CP 3                  FE 03
            5              JR Z,WWWW             23 0E
            7              CP 130                FE 82
            9              JR Z,WWWW             28 0A
            B              CP 131                FE 83
            D              JR Z,WWWW             28 06
            F              LD (IX),0             DD 36 00 00
         4123              JR UUUU               18 04
            5     WWWW:    LD (IX),128           DD 36 00 80
            9     UUUU:    INC IX                DD 23
            B              DJNZ VVVV             10 E3
            D              DEC C                 0D
            E              JR Z,SSSS             28 08
         4130              INC IX                DD 23
            2              INC IX                DD 23
            4              INC IX                DD 23
            6              JR TTTT               18 D6
            8     SSSS:    RET                   C9
            9     TEST:    CP 127                FE 5F
            B              RET M                 F8
            C              INC D                 14
            D              RET                   C9
```

Glitchoidz Report

GRA+PIX (1:4)

p. 13, right column, 2nd paragraph, last sentence should read: "If P=0 the routine will PLOT; if P=1 it will UNPLOT."

pp. 14-15 all equations with the variables 01, 02 (zero) should be rewritten as O1, O2 (letter O).

p. 16, listing 5:

add:

9010 REM ENTER FROM POLYGON/
SEGM ENT/ARC

This line does not affect the running but it changes the SYNCSUM.

Change:

9920 IF P2>2*PI THEN LET P2+P2-2
*PI*INT (P2/(2*PI))
9925 IF T2>2*PI THEN LET T2=T2-2
*PI*INT (T2/(2*PI))

"Mini-Billboard" for 8K ROM (1:5, p.2)

20 LET A(I)=(CODE(A$)*8)+7680
21 LET A$=A$(2 TO)
50 FOR Y=0 TO 7
70 IF C>=E THEN GOTO 100

Note: To use the full 8 letter capability of the program you will need additional RAM.

The PEEK Function and POKE Command (1:5, p. 22)

In the note at the bottom of the listing, 129 should be 120.

An Inventory System (1:6)

p. 30:

620 FOR B=1 TO 150
130 . . . "##UNITS."

p. 31:

1780 IF W<>1 THEN GOTO 6
1830 LET I$(B)=I$(B+1)
3068 IF C$=M$(B) THEN GOTO 3100

The Hidden Chessman (1:6, p 43)

50 LET Q=PEEK(16396)+256*PEEK(
16397)+2*X-1+34*(Y-1)

You May Fire When Ready, Gridley!

John Sampson

Have you ever wished that you could command a submarine, survey the sea through the periscope to locate the enemy fleet, and give the orders to fire your torpedos at the target ship? Well, now thanks to the ZX80 and the *Torpedo Alley* program from Zeta Software you can do just that, and for very little money.

You get this program, as you do all the Zeta programs, in typewritten format with the listings, directions, and explanations of how the program functions. This three page program has two parts: the Basic program and a machine code routine which, once entered, resides in a REM statement so that you can save and load the program with ease.

After entering the program, which is very nicely documented with comments on the right hand side of the page throughout the listing, a command of GOTO 100 produces a screen display of the view through your periscope. Each press of NEWLINE moves a destroyer from right to left across your periscope viewscreen. When you decide to fire a torpedo, you press T and NEWLINE. The torpedos leave trails through the water as they streak toward their targets. If your aim is good, the target ship explodes; if you miss, the torpedo explodes harmlessly in the water or resets the display to the next ship.

Now what would you expect to pay for such a program? $20? $10? Would you believe . . . $2? That is right! All of Zeta Software's programs are very reasonably priced. Their catalog includes utility, educational, and game programs. A minimum number of listings is required per order and $2.50 extra for shipping and handling. Most of their programs are available also on cassette for $5 additional per order. 8K ROM and 16K RAM programs are also now available.

SYNC

SOFTWARE PROFILE

Name: Torpedo Alley;
The ROM Reader
Type: Fantasy; Utility
System: ZX80
Format: Typewritten listings
Language: Basic and machine code
Summary: A lot for your money
Price: $2.00; $5.00
Manufacturer:
Zeta Software
P.O. Box 3522
Greenville, SC 29608-3522

Torpedo Alley does have two limitations which some users may feel. First, the program does not keep any score of how many ships pass, how many torpedos you fire, or how many ships are destroyed. Second, because the program uses only four random speeds for the ships, you learn very quickly which ships to fire at for a hit. But, even with these limitations in 1K, the program is very enjoyable, and some of you with 16K RAMs will, I am sure, expand the program, as I intend to do.

An example of Zeta's utility programs is *The ROM Reader* for $5. This program will, when used properly, disassemble the ROM or any machine code program. It requires a rather large amount of typing, so it should be saved several times during entry.

If you have only 1K of RAM, you will have to make several programs and use each in turn to get some of the disassembled statements each time. If you have 16K, you can make a simple change in the program and get it to return all 696 Z80A instructions. The program displays 10 bytes at a time and takes a little while to run, so you must be patient.

When you run the program, you must enter a starting address in decimal. The display will show the address of each byte in decimal, the contents of each byte in decimal, and the disassembled statement for each byte, such as, LD BC,NN or JP NN. When the content of a byte is a number which is being acted upon by the preceding instruction, then the disassembled instruction for that byte should be ignored. As I mentioned before, this program is for the serious devotee, and the average user will find it challenging. However, it is an excellent program for someone who has some understanding of how machine code works. The program has a search routine using the information in REM statements which works beautifully although it takes a while to run.

These two examples of programs illustrate the solid but inexpensive programs available from Zeta. ▪

John Sampson, 23-51 123rd St., College Point, NY 11356.

Hardware Review

MicroAce Video Upgrade

Tom Keeney

Smooth flicker free graphics has been the "impossible dream" for Sinclair ZX80 or MicroAce owners, at least until recently. It has been particularly frustrating to have the new 8K ROM and realize that this capability exists on the chip but is denied the ZX80 user! The MicroAce Video Upgrade was designed to eliminate this problem, and it works very well.

The Upgrade comes as a kit with a high quality PC board, a sack full of parts, and some instructions. These are a bit sketchy but the board layout is clear, and, if you have some exerience selecting and assembling electronic components, you should have no trouble although it is certainly not a "Heathkit."

Installing the kit is another problem. I assembled the board in less than an hour, but spent several days figuring out where to put and how to attach it. It will not fit inside the ZX80's case and some provision for mounting must be made. I placed mine in an external chassis box which also houses my keyboard beeper and connected that assembly to the computer with 14 conductor ribbon cable. The electrical problems I had with the Video Upgrade stem from the fact that I have a VHF modulator in my system. Installation procedures for these differ from those equipped with UHF modulators and the differenes are not adequately explained. After some false starts and some unnecessary board surgery, the following was determined:

a) If you have a VHF modulator do *not* make the indicated cuts between:

1) IC21 pin 1 and IC19 pin 5
2) IC19 pin 5 and R32+35

b) Do make the indicated cut in the SYNC track (IC19 pin 6 and the base resistor at TR1).

c) The indicated connection between IC21 pin 2 and R32+35 is unnecessary.

You should also be prepared to build a simple buffer circuit for the output of the Video Upgrade. The circuit for this is shown adequately in the instructions. However, you have to get the parts from Radio Shack and no mounting provisions is made on the PC board. I installed mine in the 14 pin dip socket interface.

After assembling and installing my Video Upgrade kit, I applied the power and it worked the first time although an adjustment was required to center the K vertically inside the cursor. My characters were actually split horizontally and rolled vertically so that the middle of the cursor was a space with the bottom of the K at the top and the top of the K at the bottom. The adjustment is to be expected and is adequately explained in the instructions.

In operation the Video Upgrade is great. The SLOW and FAST commands work exactly as described in the 8K ROM manual.

With the modification, the ZX80 produces pleasant, flicker free displays and smooth animation in the SLOW mode. It is, however, *awfully slow*. The only solution to this appears to be machine code graphics. The only lingering peculiarity is the fact that the top row of print is inclined slightly to the right (this occurs in the SLOW mode only; PAUSE or FAST displays are entirely normal). MicroAce says that fixing this would require a complete redesign of the ZX80 board. The distortion is, in my opinion, a *very* minor problem.

Since I installed the Video Upgrade, my software has become increasingly display oriented. Even if you are not interested in "games," the MicroAce Video Upgrade is worthwhile modification to your ZX80. It will turn it into a ZX81 for only $29. Quite a bargain!

Tom Keeney, 9629 Dortmund Dr., Huntsville, AL 35803.

Hardware Helpers

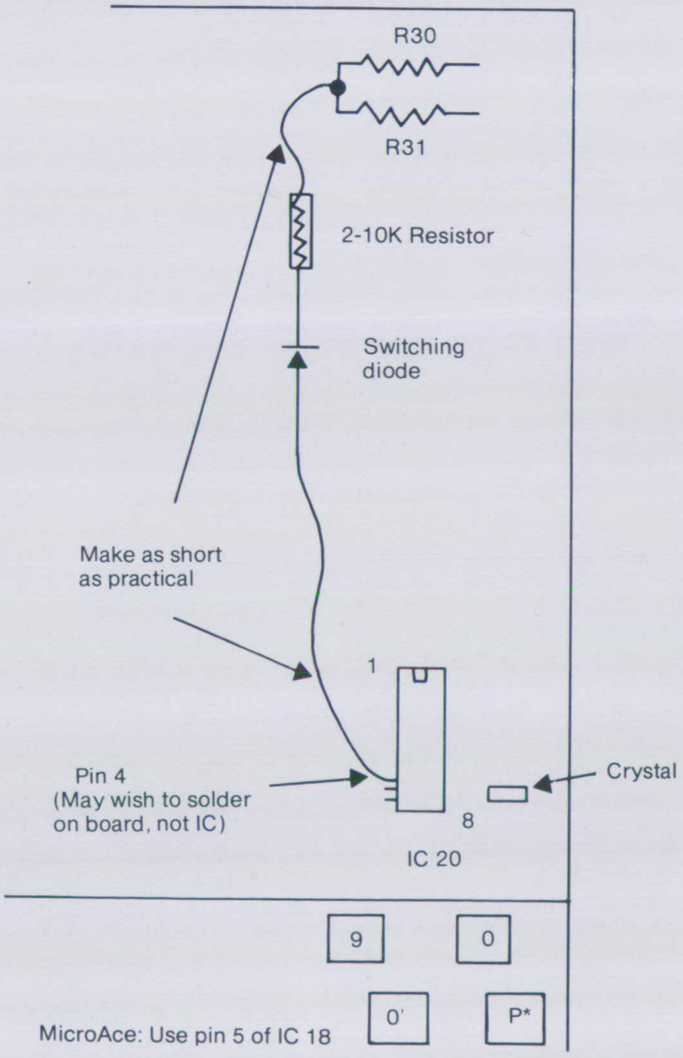
Daniel E. Schaaf

High Contrast Inverse Resolution

One disappointment I had with the ZX80 was the compromise which had to be made between resolution of inverse video characters and the contrast or sharpness of the screen. I drive a cheap tube TV set directly as a monitor and the only way to maintain readability of the cursor was to set the contrast low. The following idea helped eliminate some of that problem and opened new graphics potentials.

By placing a 5K resistor and diode between the video out (before the modulator at the junction of R30, 32) and pin 4 of IC-20 (see drawing below and schematic as published in *SYNC* 1:1) everytime an inverse character is printed a small amount of white is added to it thus lifting the inverse video out of the soup even in high contrast situations. An added bonus is that inverses are no longer exactly inverse. Graphics now have a pleasant variety of four shades of grey. The effect on some graphics is the illusion of depth or texture. Another feature is that this resistor/diode can also be used to probe the timing of events within the display field by terminating the diode at various points within the computer instead of at the pin mentioned above. A variety of masks, field patterns (mostly from the refresh cycle), and other video goodies exists within. One mask in particular, if shifted four clock cycles later, shades the text thus aiding readability. Another blocks out the main graphics area whether or not the field is occupied. The smaller the resistor's value the blacker the masks and patterns become. If too small, however, a crash is possible. The greatest use to me of this mode has been is unlocking event timing within the ZX80. ▚

Figure 2.

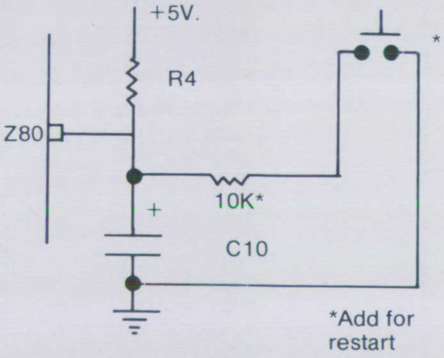


Hardware Relief from Crashes

After many a POKE where I should not have been POKEing, the ritual of unplugging the ZX80 to restart became a chore. It may seem silly, but there may be some PEEKing and POKEing programmers who do not know that a reset pin exists on the Z80 microprocessor. After several bouts with crashed programs I gave in and placed a push button switch and resistor as shown in the drawing on my ZX80. Since I drive my monitor directly I had removed the modulator long ago and put the bright red button nicely in the RF out hole. Now when programs fall into the black hole of endless, breakless loops, I press the big red, and relief is milli-seconds away. ▚

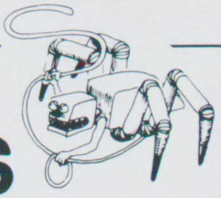
Figure 1.

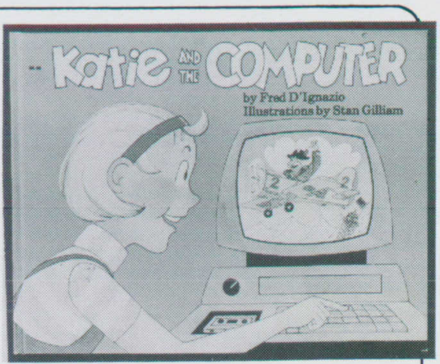
On the MicroAce, the capacitor is C2. For both computers, it may be easiest to solder to R4.



Daniel E. Schaaf, 306 N. Carroll Ave., Michigan City, IN 46360.

new friends for your child...





Katie and the Computer

Fred D'Ignazio and Stan Gilliam have created a delightful picture book adventure that explains how a computer works to a child. Katie "falls" into the imaginary land of Cybernia inside her Daddy's home computer. Her journey parallels the path of a simple command through the stages of processing in a computer, thus explaining the fundamentals of computer operation to 4 to 10 year olds. Supplemental explanatory information on computers, bytes, hardware and software is contained in the front and back end papers.

Thrill with your children as they join the Flower Bytes on a bobsled race to the CPU. Share Katie's excitement as she encounters the multi-legged and mean Bug who lassoes her plane and spins her into a terrifying loop. Laugh at the madcap race she takes with the Flower Painters by bus to the CRT.

"Towards a higher goal, the book teaches the rewards of absorbing the carefully-written word and anticipating the next page with enthusiasm..."

The Leader

"Children might not suspect at first there's a method to all this madness—a lesson about how computers work. It does its job well."

The Charlotte Observer

"...the book is both entertaining and educational."

Infosystems

Order Today

Katie and the Computer is hardbound, illustrated in full color throughout and costs just $6.95. A T-Shirt picturing the program bug in the story is also available (purple bug on a beige shirt). Shirts are available in adult S, M, L, XL, children's S, M and L and cost just $6.00.

To order send payment plus $2.00 postage and handling per order to the address below.

creative computing

Dept. Z0121
One Park Avenue, Room 458
New York, N.Y. 10016

*Residents of CA, CO, DC, FL, IL, MA, MI, MO, NJ, NY State, OH, SC, TN, and VT, add applicable sales tax.

Computer Lawnmower

Can a computer mow your lawn? Not yet. But a flowchart can show you how to make money cutting five lawns a day. The flowchart is easy. Mowing the lawns is still hard work.

Dr. Sylvia Charp and Marion Ball wanted a way to introduce basic computer concepts to children in grades 5 to 9 of the Philadelphia City Schools. So they identified some tasks that kids understood like mowing lawns, issuing paychecks and controlling traffic lights. They showed how computers are used in these tasks.

Flowcharts - A basic concept

They devised flowcharts. They located scores of photos. And they found an artistic high school student to illustrate these concepts with lively full-color drawings.

They then wrote a light-hearted but informative text to tie it all together. It talked about kinds of computers, what goes on inside the machine, the language of the computer and how computers work for us.

They took the problem of averaging class grades and showed how a simple yrogram could be written to do this job.

Well-qualified authors

Marion Ball has written other books on computer literacy. Sylvia Charp is the director of educational compuuting for Philadelphia City Schools. They pooled their talents to produce this book, Be A Computer Literate.

This easy-to-read book explains how computers are used in medicine, law enforcement, art, business, transportation and education. It's interesting and understandable.

Too much demand

The Bell System distributed 50,000 copies to schools throughout the U.S. but they couldn't meet the continuing demand. So Creative Computing Press now distributes the book. Order yours today. If, after reading it, you do not feel that you are "computer literate," return it for a full refund.

Send $3.95* plus $1.00 postage and handling to Creative Computing, Dept. Z0120, One Park Avenue, Room 458, New York, NY 10016.

creative computing

*Residents of CA, CO, DC, FL, IL, MA, MI, MO, NJ, NY State, OH, SC, TN, and VT, add applicable sales tax.

8K ROM
1K RAM

Part 1

Understanding Floating-point Arithmetic

Ian Logan

The aim of this article is to give the reader some insight into the complex world of floating-point arithmetic. Since the 4K ROM provided only integer arithmetic, readers who possess only this ROM will be unable to try the programs. Nevertheless they will be able to follow the text.

In the Sinclair Manual, *ZX81 Basic Programming*, chapter 27, Steven Vickers shows that a floating-point number consists of a single exponent byte and 4 mantissa bytes, but he gives no further information. In order to understand this subject it is probably best to return to first principles—so with pencil and paper to hand proceed.

Decimal format

In the beginning there were only simple integers. But soon they begat decimal numbers, which have an integer part, a decimal-point and a decimal part. And in their turn decimal numbers begat E-format, which has a mantissa part, an 'E' and an exponent part.

For example, the number 'four' can be expressed as:

4	- its integer value
4.000	- its decimal value
40000E-4	- just one of many E-format choices

It can readily be seen that in the E-format we have the essential parts of floating-point notation for decimal numbers all given, but it is useful at this point to introduce two conventions that will help us in conversion from decimal-floating-point to binary-floating-point.

1) Always express the mantissa starting with the decimal-point.

2) Do not attribute a sign to the mantissa. Simply state whether the value is positive or negative. So instead of:

	Write:	
40000E-4	.4E1	& positive
0.00678	.678E-2	& positive
-223.9	.2239E3	& negative
-0.7	.7E0	& negative

These conventions can be considered to be 'normalizing' the floating-point decimal number.

With a decimal number in its 'normalized' form we can now state that the mantissa is the decimal part of the form and the exponent is the integer part after the 'E'. The exponent is a signed integer and the overall form is either positive or negative. Consider the examples in Figure 1. The reader is urged to try further examples. (Perhaps with a friend marking the results.)

Figure 1.

Decimal	Normalized	Exponent	Mantissa	+/-
4	.4E1	+1	4	+
40	.4E2	+2	4	+
.4	.4E0	+0	4	+
-40.0	.4E2	+2	4	-
-123.456	.123456E3	+3	123456	-

Binary Format

As the 8K ROM program deals with binary-floating-point numbers and not decimal-floating-point numbers, the reader will now have to convert the above conclusions so that they apply to binary-format numbers.

First, consider the state when all binary numbers represented integer values, that is:

Decimal	Binary
45	0010 1101
255	1111 1111

In this state all values are integers and positive only. Next consider fixed-point binary numbers in which there is a fixed binary-point separating the integer byte(s) from the fraction bytes(s). That is:

Decimal Form	Binary Form		
	integer	point	fraction
45	0010 1101	•	00000000
45.5	0010 1101	•	10000000
45.75	0010 1101	•	11000000
45.875	0010 1101	•	11100000

Note that in a fixed-point number the first bit after the binary-point represents the value .5 and the second bit .25 etc. (The values diminish by a factor of 2.)

However, it is also possible to consider the fraction part byte by byte, which in decimal can be illustrated as follows:

From above, .11100000 gives 224/256 as the fraction part and this does give 0.875.

Now at last the binary numbers can be

Ian Logan, 24 Nurses Lane, Skellingthorpe, Lincoln LN6 0TT.

'normalized.' All that needs to be done is for the whole number to be moved to the left, or the right, as needed so that the most significant bit comes to be the first bit of the fraction part. The exponent is then given as the number of moves made (+ right, - left) and the mantissa is the number of bits wanted from the fraction part.

Hence from above:

Decimal Form	**Exponent**	**Mantissa**
45	+6 (dec.)	10110100
45.875	+6 (dec.)	10110111 (a bit is lost)

Note that in the example with a mantissa being limited to just 8 bits that the values 45.75 and 45.875 cannot be distinguished. This shows why the 8K ROM uses not one but 4 bytes for the mantissa and even then it 'rounds' off values—sometimes inconveniently.

But how are negative numbers dealt with? Well, it is easy; there is just a statement made to say whether the value is positive or negative. For example:

Decimal Form	**Exponent**	**Mantissa**	**+/-**
255	+8 (dec.)	11111111	+
-255	+8 (dec.)	11111111	-

Now it is time to run Program 1. This *Floating-point Demonstration Program* asks the user to enter any decimal number that he may wish, including fraction parts and 'E's '. The program then returns the true exponent, e', and the four bytes of the mantissa. (e' is the exponent as developed above.) For example, entering the number 255 gives:

Decimal number	255
Its exponent	8
And mantissa	255 0 0 0 0
And it is	POSITIVE

and entering -9.9E37 will give:

Decimal number	-9.9E+37
Its exponent	127
And mantissa	148 245 105 108
And it is	NEGATIVE

Note: The last value can be checked by trying the line:

```
PRINT (148/256+245/256**2+105/25
6**3+180/256**4)*2**126*2
```

which gives 9.9E+37 as expected. (Note that 2**126*2 is used to prevent overflow.) Program 1 works by reading the floating-point number that has been attributed to the variable A as that number occurs in the variable area of the RAM. Certain changes have to be made to these bytes in order to give the true exponent and the appropriate mantissa. Note for interest the differences between values of A that ought to be the same. See Figure 2. The later result is a 'rounding' error.

Whereas Program 1 borrows the result of the ROM program to get to its answer, Program 2, *A Floating-point Builder*, develops the result by successive multiplications, divisions, and subtractions. So try Program 2 in order to become more familiar with binary floating-point numbers.

Figure 2.

	1/2	dec.	gives	Exp. 0	Mantissa 128 0 0 0
but	.5	dec.	gives	Exp. -1	Mantissa 255 255 255 255

Note: The lines 170, 180, and 210 are all attempts to get around the problem of 'rounding' errors. However, the serious reader might be interested in the fact that with an initial value of A such as 8 then the value of A at line 170 is:

$.999999999 < A < 1$

'PRINT A' gives 1, but 'IF A=1' is false. The explanation lies in the fact that A has the binary value of:

EXP. 0 , Mantissa 127 255 255 253

instead of the expected

EXP. 1 , Mantissa 128 0 0 0

and therefore shows that the COMPARISON operation is of greater sensitivity than the PRINT operation.

Does this 'bug' account for some programming problems?

Sinclair floating-point conventions

So far in this article I have described the use of the true exponent and the true mantissa, but in Sinclair machines the floating-point numbers follow two conventions which are:

1) The exponent byte always has 128 decimal, Hex.80, added to it, unless it is the exponent for the value zero when the exponent is always zero. Hence the 'augmented exponent,' e, is the 'true exponent,' e', +128. (See how in line 120 of Program 1 this is taken into account.)

2) The true numeric bit 7 of the first byte of the mantissa which is always set in a floating-point that has been 'normalized' is understood to be present and the bit replaced by a sign-bit. This bit is set for negative numbers and reset for positive numbers (and zero). (See how in line 140 of Program 1 this is taken into account.)

To make this clear consider the examples in Figure 3.

Figure 3.

Decimal Format	**True Format**		**Sinclair Format**	
	Exp.	**Mant.**	**Exp.**	**Mant.**
1.0	1	128 0 0 0	129	0 0 0 0
2.0	2	128 0 0 0	130	0 0 0 0
-2.0	2	128 0 0 0	130	128 0 0 0
3.0	2	192 0 0 0	130	64 0 0 0
-3.0	2	192 0 0 0	130	192 0 0 0
0.0	0	0 0 0 0	0	0 0 0 0

Conclusions

Floating-point notation is logical, tedious perhaps, but very useful.

By way of lighter relief this month's game is an example of Basic programming that shows how bytes can be saved in 8K ROM programs—who said the 8K ROM wastes bytes?

The idea of the game is simply to find a number that results in the pattern filling the whole board. My best score so far is about 100.

Remember that RND generates a given series of numbers, depending on the SEED for its starting point, but additional dummy calls to RND will create new series. E.g.,

```
145 POKE 0,RND
```

would be economic for a simple arithmetic series—alternate calls to RND are used by the 'pattern.'

Part 2 of "Understanding Floating-point Arithmetic" will discuss the third language of the 8K ROM—the Calculator Language.

Bibliography

Sinclair ZX81 ROM Disassembly, Part A: 0000 H-00F54 H, by Dr. Ian Logan. Melbourne House outlets—£7. (Deals with the 'operating system' part of the 8K ROM program).

Sinclair ZX81 ROM Disassembly, Part B: 0F55 H-1DFE H, by Dr. Ian Logan and Dr. Frank O'Hara. Melbourne House outlets—£8. (Deals with 'expression evaluation' and the 'calculator routines' in full detail). ▚

Program 1: Floating-point Demonstration Program

```
  10 PRINT AT 17,0;"ENTER ANY NU          Any decimal number.
MBER"
  20 INPUT A
  30 CLS
  40 LET V=PEEK 16400+256*PEEK 1          Get the present value of
6401                                      VARS.
  50 DIM B(5)                             For the 5 bytes.
  60 FOR C=1 TO 5                         Get each byte from the
  70 LET B(C)=PEEK (V+C)                  variable area.
  80 NEXT C
  90 PRINT "DECIMAL NUMBER";TAB
17;A
 100 PRINT
 110 PRINT
 120 PRINT "ITS EXPONUNT";TAB 17
;B(1)-128*(B(1)<>0)                       Form the true exponent.
 130 PRINT
 140 PRINT "AND MANTISSA";TAB 17
;(A<>0)*(B(2)+128*(B(2)<128));TA          Form the true mantissa.
B 21;B(3);TAB 25;B(4);TAB 29;B(5
)
 150 PRINT
 160 PRINT "AND IT IS";TAB 17;"P          Give the sign.
OSITIVE" AND (A>=0);"NEGATIVE" A
ND (A<0)
 170 RUN
```

Program 2: Floating-point Builder

```
  10 INPUT A                              Any decimal value.
  20 CLS
  30 LET B=SGN A                          Keep the sign.
  40 PRINT "DECIMAL NUMBER";TAB
17;A
  50 LET A=ABS A                          Ignore negative sign.
  60 PRINT
  70 PRINT
  80 LET E=0                              Set exponent to zero.
  90 PRINT "ITS EXPONENT";TAB 17
;E
 100 IF A>=.5 AND A<=1 OR A=0 TH          Exit when "normalized."
EN GOTO 150
 110 LET E=E-(A<1)+(A>1)                  Exponent changes by one.
 120 LET A=A*(.5+1.5*(A<1))               A chanes by .5 or 2 fold.
 130 PRINT AT 3,17;E                      Watch it changing in SLOW.
 140 GOTO 100
 150 PRINT
 160 PRINT "AND MANTISSA";TAB 17
;
 170 IF A>.999999999 THEN LET A=
1                                         See text.
 180 LET F=.003906249997                  A little under 1/256.
 190 FOR G=1 TO 4                         Each mantissa byte.
 200 LET H=INT (A/F)                      The decimal value.
 210 IF H>255 THEN LET H=128              For a rounding error.
 220 PRINT H;" "                          The byte and a "space."
 230 LET A=A-INT (A/F)*F                  Decrease A.
 240 LET F=F/256                          Change for each byte.
 250 NEXT G
 260 PRINT
 270 PRINT
 280 PRINT "AND IT IS";TAB 17;"P          Fetch the sign.
OSI" AND (B>=0);"NEGA" AND (B<0)
;"TIVE"
 290 RUN
```

Program 3: Floating-point Number Game

```
  10 PRINT AT VAL "20",NOT PI;"N
EW NUMBER?"
  20 INPUT N
  30 RAND N
  40 CLS
  50 FOR A=NOT PI TO VAL "15"
  60 PRINT "█";"██████████████"
AND (NOT A OR A=VAL "15");TAB VA
L "15";"█"
  70 NEXT A
  80 LET A=VAL "7"
  90 LET B=A
 100 LET C=NOT PI
 110 LET D=VAL "30"
 120 LET D=D-SGN PI
 130 IF D=NOT PI THEN RUN
 140 LET E=INT (RND*INT PI)-SGN
PI
 150 LET F=INT (RND*INT PI)-SGN
PI
 160 PRINT AT A+E,B+F;
 170 IF PEEK (PEEK VAL "16398"+V
AL "256"*PEEK VAL "16399")<>NOT
PI THEN GOTO VAL "120"
 180 PRINT "*"
 190 LET C=C+SGN PI
 200 PRINT AT VAL "18",NOT PI;"S
TARS = ";C
 210 LET A=A+E
 220 LET B=B+F
 230 GOTO VAL "110"
```

8K ROM
1K RAM

Linear Regression

Jon T. Passler

The "Linear Regression" Program computes the linear relationship between two sets of variables, expressed as the linear regression equation, and calculates the coefficient of determination, an indicator of the strength of the relationship. Given a set of two variables labelled X and Y, the program will yield an equation describing Y as a function of X.

These variables can be taken from any situation in which a logical relationship is expected, such as rainfall and crop yield, the prime interest rate and auto sales, or time and any variable which changes (generally in one direction) over a period of time. For a time series, X can be expressed in periods, starting with period 1.

The coefficient of determination, R2 or R squared, is a measure of how much of the variability in Y is "explained" by, or related to, the variability in X. R2 varies between 0 and 1, and R2 multiplied by 100 gives a percent indication of the validity of, or accuracy in, expressing Y as a function of X.

For a quick example, let Y = 1 + 2 * X. If X = 1, 2, and 3, then Y would be 3, 5, and 7. Run the program, enter a 3 in response to the number of entries, then enter X's and Y's pairwise, or, to mix things up a bit, enter 2, 5, 3, 7 and 1, 3. You should get the equation Y = 1 + 2 * X back, and an R2 of 1, or 100%, since the equation perfectly describes the relationship between each pair of entries.

Linear regression can be used to approximate the value of one variable (given the value of another), identify the trend in time series and forecast future values, or evaluate the influence of one variable on another (R2). ▚

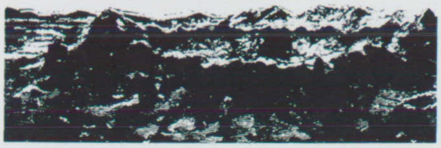
Jon T. Passler, 344 Cabot St., Beverly, MA 01915.

```
  10 REM LINEAR REG
  20 PRINT "N OF ITEMS?"
  30 INPUT N
  40 LET SX=0
  50 LET SY=0
  60 LET XX=0
  70 LET YY=0
  80 LET XY=0
  90 PRINT "INPUT X/S AND Y/S"
 100 FOR A=1 TO N
 110 IF A>20 THEN SCROLL
 120 INPUT X
 130 PRINT X;"  ";
 140 INPUT Y
 150 PRINT Y
 160 LET SX=SX+X
 170 LET SY=SY+Y
 180 LET XX=XX+ABS X**2
 190 LET YY=YY+ABS Y**2
 200 LET XY=XY+X*Y
 210 NEXT A
 220 CLS
 230 PRINT
 240 PRINT "Y = ";(XX*SY-SX*XY)/
(N*XX-ABS SX**2);" + ";(N*XY-SX*
SY)/(N*XX-ABS SX**2);" * X"
 250 PRINT
 260 PRINT "R2 = ";ABS ((N*XY-SX
*SY)/((N*XX-ABS SX**2)**.5*(N*YY
-ABS SY**2)**.5))**2
```

8K ROM
1K RAM

Lunar Lander

Chuck Dawson



You are the Command Pilot of the *Lunar Lander SYNC*. You are now in the final stages of your descent and you must make your landing before your fuel runs out. You select the thrust settings from your keyboard control system by pressing a key from 1 to 10. The computer does not wait for you to think because the law of gravity operates and the lander continues its descent. If you hit the surface at a velocity of more than 100 feet per second, you will collapse your landing gear and crash on the desolate, rock strewn surface. Once a thrust setting is chosen, it stays set until you choose another. You can cut the engines altogether by pressing zero. The engines also stop when you run out of fuel. Naturally, this is also a crash landing. When setting the power, hold down the key for a full cycle (one blink to the next) so that you are sure your key has been read by step 11.

In this game the screen display shows the rugged lunar surface at the bottom. Your instrument panel is on the right. You have vertical velocity, altitude, and fuel remaining. The 1K program just barely fits into the 1K with no room to spare. Use the one and two digit line numbers as shown, and do not add any remarks. If you have more memory, you can finish out the surface to the right edge of the display. The 2K program adds possibilities.

If you want to change the level of difficulty, change the V in line 39. The program takes advantage of the ZX81's INKEY$ feature for game input. Be sure to enter the spaces in the PRINT statements in both versions very carefully. You can refer to the lines above and below for the spaces. The listings are a direct printout from the ZX81 and show the screen as it should look.

Two or more can play *Lunar Lander* with the winner being the player with the most fuel left. A crash is disqualifying, of course.

Go ahead, you're GO FOR LANDING.

Notes:

1K Version:

20 Graphics: Inverse O; T, 6, Y; 5, 6, 8; R, 7, 6, 7, E.

37 Graphics: 9 graphic A, alternated with 8 graphic D.

2K Version:

29 Graphics: same as 1K line 20.

37 Graphics: 14 graphic A, alternated with 13 graphic D. ▚

Chuck Dawson, 6520 Victoria, Ft. Worth, TX 76118.

Program 1: Lunar Lander: 8K ROM; 1K RAM

```
   1 LET V=0
   3 LET H=1500
   5 LET A=0
   7 LET F=7000
   9 GOTO 15
  11 IF INKEY$<>"" THEN LET A=VA
L INKEY$
  13 IF NOT F THEN LET A=0
  15 LET V=V+10*A-32
  17 LET H=H+V
  19 IF H<50 THEN LET H=0
  21 LET L=INT (H/100+.5)
  23 LET F=F-100*A
  25 IF F<0 THEN LET F=0
  27 CLS
  29 PRINT AT 15-L,L;"  ▣";TAB L
;" ▟▄▙";TAB L;" ▐▄▌";TAB L;"▀▛▀▜
▛"
  31 IF A THEN PRINT TAB L;"  *"
  33 PRINT AT 12,15;"VEL  ALT  F
UEL"
  35 PRINT TAB 14;V;TAB 20;H;TAB
 25;F
  37 PRINT AT 19,0;"▒▄▒▄▒▄▒▄▒▄▒▄
▒▄▒▄▒"
  39 IF NOT H AND V<-100 THEN PR
INT "*CRASH*"
  41 PAUSE 60
  43 POKE 16437,255
  45 IF H THEN GOTO 11
```

Program 2: Lunar Lander: 8K ROM; 2K RAM

```
   1 LET V=0
   3 LET H=1500
   5 LET F=7000
   7 LET A=0
   9 LET L=15
  10 GOTO 15
  11 IF INKEY$<>"" THEN LET A=VA
L INKEY$
  13 IF NOT F THEN LET A=0
  14 LET V=V+10*A-32
  15 IF V>0 THEN GOSUB 70
  16 IF V<-215 THEN LET V=-215
  17 LET H=H+V
  19 IF H<50 THEN LET H=0
  20 LET C=L
  21 LET L=INT (H/100+.5)
  23 LET F=F-100*A
  25 IF F<0 THEN LET F=0
  27 PRINT AT 15-C,C+2;"  ";TAB C
;"    "
  29 PRINT AT 15-L,L;"  ▣    ";TA
B L;" ▟▄▙    ";TAB L;" ▐▄▌    ";TA
B L;"▀▛▀▜▛"
  30 IF A THEN PRINT TAB L+2;"*"
  31 PRINT AT 10,15;"THROTTLE -
";A
  33 PRINT AT 12,15;"VEL  ALT
FUEL"
  35 PRINT TAB 14;V;"   ";TAB 20;
H;"   ";TAB 26;F;"   "
  37 PRINT AT 19,0;"▒▄▒▄▒▄▒▄▒▄▒▄
▒▄▒▄▒▄▒▄▒▄▒▄▒▄▒"
  39 IF NOT H AND V<-100 THEN PR
INT "**CRASH**"
  41 IF H THEN GOTO 11
  43 STOP
  70 PRINT AT 17-L,L;"        ";TA
B L;"     ";TAB L;"    "
  72 RETURN
```

Sample Run

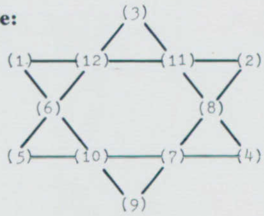
```
          VEL   ALT   FUEL
         -52    718   2400
```

puzzle answers

A Rectangular Problem: The answer is 51 rectangles.

Who's Who: Tom is married to Nancy, Joe is married to Mary, and Fred is married to Patty. Hint: Nancy played with Fred against her husband.

The "26" Puzzle:

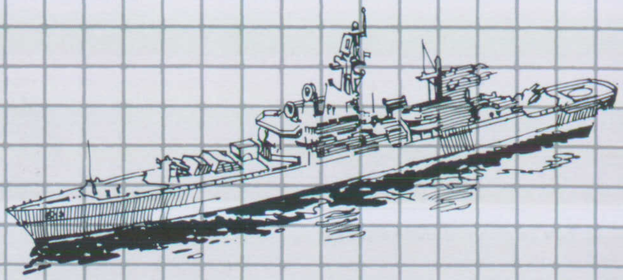


A Word Square Problem:

```
NESTLES
ENTRANT
STRANGE
TRAITOR
LANTERN
ENGORGE
STERNER
```

The Flock of Geese: The number of the flock was 36. For, taking the lowest number (4), which is divisible by 2 and by 4 (as, from the conditions of the problem, it is clear that the required number must be), and going through the process suggested with such number, we have the following result: 4+4(as many more) +2(half as many more) + 1(one-fourth as many more) =11. Dividing 99(the total to be obtained after going through the same process with the actual number in the flock) by the number thus obtained, we find the quotient to be 9. 4, therefore, multiplied by 9 (=36) should be the required number. Putting it to the test, we find that 36+36+18+9=99, exactly answering the conditions.

4K ROM
1K RAM



Battleship Solitaire

Bob Dusenberry

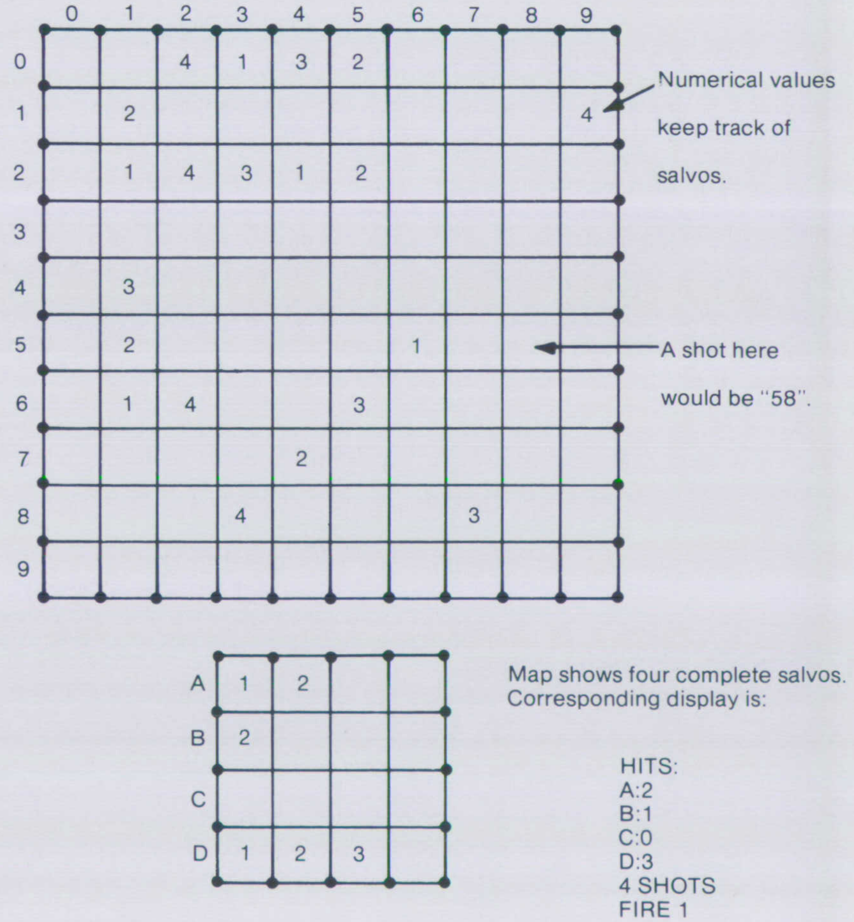
Battleship was a game we used to play as kids (and later)—with pencil and paper before the toy manufacturers plasticized it. You remember—each of you placed some number of "ships" of various sizes in a coordinate grid "ocean" and then took "shots" at each other by calling out coordinate locations in turn. After each "salvo," hits were reported and recorded. Loss of a certain one of your ships would penalize you, say, two shots off the next salvo. The object was to wipe out your opponent before he got you.

This *Battleship Solitaire* program allows you to play a similar game against the computer, but with the modification that you do all the shooting. Even though you have no ships to be sunk, you are still subject to penalties. Each time you fail to score at least one hit per salvo you lose one shot. The object of this game is to sink all enemy ships before you are out of shots.

As listed, the *Battleship Solitaire* program provides a 10 by 10 ocean with numerical coordinates. The "tens" digit appears on the vertical axis with the "units" digit on the horizontal axis. You will be seeking four ships of four boxes or segments each. They may be placed horizontally, vertically, or on either diagonal. Since both the bow of the ship and its orientation are determined by the ZX80 on a random basis, a new situation is added to the game: ships may abut, cross, or even share the same coordinate location! (What do you want from a 1K program?) If the concept of intersecting battleships offends you, let some be submarines at different depths on the same X-Y coordinate.

The program has three distinct phases. First, the four ships are randomly—and secretly—placed in the ocean. Second is your turn. You are advised of the number of shots per salvo (initially 5), which you take using coordinate designations. At the end of each salvo the ZX80 reports the accumulated number of "hits" on each of the four ships: A, B, C, and D. It also displays the number of remaining shots for your next salvo, taking into account your hits and misses. Eventually it will announce "YOU WIN" or "YOU LOSE". During the third phase of the program, accessible on a GO TO basis after the battle is over, the ZX80 generates a display of the ocean complete with those elusive ships.

Figure 1. Mapping your shots.



	0	1	2	3	4	5	6	7	8	9
0			4	1	3	2				
1		2								4
2		1	4	3	1	2				
3										
4		3								
5		2					1			
6		1	4			3				
7					2					
8				4				3		
9										

Numerical values keep track of salvos.

A shot here would be "58".

A	1	2		
B	2			
C				
D	1	2	3	

Map shows four complete salvos. Corresponding display is:

```
HITS:
A:2
B:1
C:0
D:3
4 SHOTS
FIRE 1
```

Bob Dusenberry, 77 Moraine Rd., Morris Plains, NJ 07950.

Before running the program it is recommended that pencil and paper be available for you to record your shots and hits. By drawing a 10 by 10 grid plus a separate set of four-box diagrams, one for each ship, you can keep track of shots you have taken and ships you have hit. Big hint: mark your shots and hits of the first salvo using 1's, the second salvo using 2's, etc. Figure 1 shows such a diagram as it might appear part way through a game.

Start in the usual way, pressing RUN and NEWLINE. After the computer deploys the ships, unseen by you, it will display:

```
5 SHOTS
FIRE 1
```

You respond by pressing a two digit number, YX, representing your chosen coordinate. (Y is the vertical coordinate, X is the horizontal.) Follow with NEWLINE. The display now shows:

```
5 SHOTS
FIRE 2
```

Continue firing the rest of the salvo in the same manner. After you "Fire 5" the display will show, say:

```
HITS:
A:0
B:0
C:1
D:0
5 SHOTS
FIRE 1
```

Note that ship C has been hit, as an example, and thus the next salvo is allowed to continue with 5 shots. If all ships had come up "0", the display would show:

```
HITS:
A:0
B:0
C:0
D:0
4 SHOTS
FIRE 1
```

You have a one-shot penalty for your failure to hit anything on the first salvo. "Four" is now the maximum number of shots per salvo during this game.

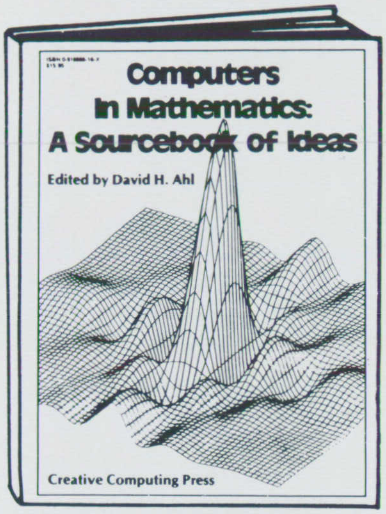
Remember that the number of hits displayed is accumulative. Thus "C:4" would indicate that ship C has been sunk. If you should hit a coordinate shared by two overlapping ships, both ships will be scored as "hit".

The game continues, with the number of hits increasing (hopefully) and the number of shots decreasing (woefully) until the display shows:

```
HITS:
A:4
B:4
C:4
D:4
    YOU WIN
```

Sourcebook of Ideas

Many mathematics ideas can be better illustrated with a computer than with a text book.



Consider Baseball cards. If there are 50 cards in a set, how many packs of bubble gum must be purchased to obtain a complete set of players? Many students will guess over 1 million packs yet on average it's only 329.

The formula to solve this problem is not easy. The computer simulation is. Yet you as a teacher probably don't have time to devise programs to illustrate concepts like this.

Between grades 1 and 12 there are 142 mathematical concepts in which the computer can play an important role. Things like arithmetic practice, X-Y coordinates, proving geometic theorems, probability, compounding and computation of pi by inscribed polygons.

Endorsed by NCTM

The National Council of Teachers of Mathematics has strongly endorsed the use of computers in the classroom. Unfortunately most textbooks have not yet responded to this endorsement and do not include programs or computer teaching techniques. You probably don't have the time to develop all these ideas either. What to do?

For the past six years, *Creative Computing* magazine has been running two or three articles per issue written by math teachers. These are classroom proven, tested ideas complete with flowcharts, programs and sample runs.

Teachers have been ordering back issues with those applications for years. However, many of these issues are now sold out or in very short supply.

So we took the most popular 134 articles and applications and reprinted them in a giant 224-page book called *Computers in Mathematics: A Sourcebook of Ideas.*

Ready-to-use-material

This book contains pragmatic, ready to use, classroom tested ideas on everything from simply binary counting to advanced techniques like multiple regression analysis and differential equations.

The book includes many activities that don't require a computer. And if you're considering expanding your computer facilities, you'll find a section on how to select a computer complete with an invaluable microcomputer comparison chart.

Another section presents over 250 problems, puzzles, and programming ideas, more than are found in most "problem collection" books.

Computers in Mathematics: A Sourcebook of Ideas is edited by David Ahl, one of the pioneers in computer education and the founder of *Creative Computing.*

The book is not cheap. It costs $15.95. However if you were to order just half of the back issues from which articles were drawn, they would cost you over $30.

Satisfaction Guaranteed

If you are teaching mathematics in any grade between 1 and 12, we're convinced you'll find this book of tremendous value. If, after receiving it and using it for 30 days you do not agree, you may return it for a full refund plus your return postage.

To order, send $15.95* plus $1.00 postage and handling to Creative Computing, Dept. Z0122, One Park Avenue, Room 458, New York, N.Y. 10016. Orders may also be charged to your Visa, MasterCard or American Express account—by mail include credit card name, number, and expiratioin date, or if you prefer, use our 24 hour toll free number, 800-345-8112. In PA only 800-662-2444. Institutional purchase orders should include an additional $1.00 billing fee.

creative computing

*Residents of CA, CO, DC, FL, IL, MA, MI, MO, NJ, NY State, OH, SC, TN, and VT, all applicable sales tax.

On the other hand, if you run out of shots first, it will show, say:

```
HITS:
A:3
B:2
C:4
D:3
      YOU LOSE
```

The program run can be halted at any time before the end by entering "100" in place of a coordinate.

Figure 2. Typical ship deployment displays.

```
  0 1 2 3 4 5 6 7 8 9
0   X
1   X
2   X       X X X X
3   X
4               X
5             X
6 X         X
7   X     X
8     X
9       X

  0 1 2 3 4 5 6 7 8 9
0
1         X         X
2           X     X
3     X       X X
4     X       X X
5     X X
6     X   X
7           X
8
9
```

If, now that the game is over, you are curious about the ship deployment, press NEWLINE, GO TO 700, and NEWLINE to obtain a display of the ocean and the ship deployment. Be patient, it takes something over a half minute to perform this function. Figure 2 shows typical deployment displays.

Should you want to change the degree of difficulty of the game, the easiest way is to alter the initial number of shots per salvo. Fewer shots make it harder, more shots make it easier. Five shots, as programmed, seem about right, and one shot, more or less, will significantly change the difficulty. To make the change, simply edit line 300 of the program:

300 LET N=5

to include the desired value of N.

Other parameter variations were tried, such as the number of ships and ships of different sizes (e.g., five ships of 1, 2, 3, 4, and 5 segments). But in the end we returned to the four ships of four segments each.

It should be pointed out that, smart as it is, the program cannot tell if you shoot the same coordinate more than once. You are not supposed to do this, but if you happen to and score another hit on a coordinante, you will get credit for an extra "hit." That could lead to a false "WIN." Not to worry—no self-respecting solitaire player would cheat in this manner. ▣

Figure 3. Battleship Solitaire Listing

```
 10 DIM B(16)                                 Four-ship array.
 20 LET P = 1                                 Preset array index.
 40 FOR J = 1 TO 4                            Deploy ships, per ship
 60 LET D = RND(4)                            Random orientation.
 70 LET X = RND(7)                            70,80 Random X loc., modify
 80 IF D = 4 THEN LET X = RND(10)                   for vertical ship.
 90 LET Y = RND(7)                            90-100 Random Y loc., modify
100 IF D = 2 THEN LET Y = RND(10)                   for horizontal ship.
110 IF D = 1 THEN LET Y = Y + 3               Shift Y if ship slants up.
120 LET W = 1                                 120,130 Set horiz. increment
130 IF D = 4 THEN LET W = 0                         factor.
140 LET Z = 1                                 140-160 Set vert. increment
150 IF D = 2 THEN LET Z = 0                         factor.
160 IF D = 1 THEN LET Z = -1
165 LET K = 0                                 Preset segment index.
170 FOR I = P TO P + 3                        Deploy ship, per segment.
180 LET B(I) = (Y + K * Z) * 10 + X + K * W   Form coords; load array.
185 LET K = K + 1                             Increment segment index.
190 NEXT I
200 LET P = P + 4                             Increment array index.
210 NEXT J
300 LET N = 5                                 Preset no. of shots per salvo.
310 LET C = 0                                 310-340 Preset ship hits.
320 LET D = 0
330 LET E = 0
340 LET F = 0
343 LET G = 0                                 Preset salvo hits.
345 FOR J = 1 TO N                            Salvo firing, per shot.
347 PRINT N; " SHOTS"                         347,350 Print salvo info.
350 PRINT "FIRE "; J
360 INPUT A                                   Display; input shot coords.
370 IF A = 100 THEN STOP                      Optional run halt.
380 CLS
400 FOR I = 1 TO 16                           Hit check, per segment.
410 IF NOT A + 11 = B(I) THEN GO TO 460       Skip ship hits on miss.
420 IF I<5 THEN LET C = C + 1                 420-450 Increment ship hits
430 IF I>4 AND I<9 THEN LET D = D + 1               as req'd.
440 IF I>8 AND I<13 THEN LET E = E + 1
450 IF I>12 THEN LET F = F + 1
455 LET G = 1                                 Increment salvo hit.
460 NEXT I
470 NEXT J
475 PRINT "HITS:"                             475-570 Print salvo results.
480 PRINT "A:"; C
510 PRINT "B:"; D
540 PRINT "C:"; E
570 PRINT "D:"; F
600 IF C>3 AND D>3 AND E>3 AND F>3 THEN       All ships sunk, skip to end.
          GO TO 640
610 LET N = N - 1 + G                         Modify no. of shots per salvo.
620 IF N = 0 THEN GO TO 660                   Out of shots, skip to end.
630 GO TO 343                                 Recycle if not end.
640 PRINT, "YOU WIN"                          640-670 Print outcome & stop.
650 STOP
660 PRINT, "YOU LOSE"
670 STOP
700 PRINT " 0123456789"                       Print X coord. heading.
710 FOR K = 1 TO 10                           Form display, per Y coord.
720 PRINT K - 1;                              Print Y coord.
730 FOR L = 1 TO 10                           Form display, per X coord.
740 LET A$ = " "                              Preset character to blank.
760 FOR I = 1 TO 16                           Scan ship array, per segment.
770 IF B(I) = 10 * K + L THEN LET A$ = "X"    Set char. to X if coord match.
780 NEXT I
790 PRINT A$;                                 Print character.
810 NEXT L
820 PRINT ""                                  Terminate line.
830 NEXT K                                    Display after last K.
```

4K(8K) ROM
1K RAM

The ZX80 as a Cipher Machine

James John Hollandsworth

Would you believe that you can turn your ZX80 into a code machine? It can translate messages into gibberish that would take an expert hours to solve, yet a fellow ZX80 user could translate back into English in a few seconds.

You can do this by using the Vigenere cipher. This is probably the most famous cipher of all time, and is named for Blaise de Vigenere, the Frenchman who first described it in 1586. According to legend, he called it *le chiffre indechiffrable*, the indecipherable cipher. By this he meant that if a secret message using this method of encoding fell into unfriendly hands, it would be impossible for the enemy to break it. Although today it can be broken, the process is hard, long, and tedious. It can be done only by experts in cryptology who have long messages to work with. Such personalites as Lewis Carroll and Sir Admiral Francis Beaufort were interested in this cipher. The Confederate Army used it extensively. Although the military does not use the cipher today, it remains the basis of some ciphers.

The Vigenere cipher was one of the first polyalphabetic ciphers, as opposed to monoalphabetic ciphers used since the time of Caesar. In a monoalphabetic cipher, each letter in the message is replaced by a unique letter in the cipher; e.g., A may stand for F, D may stand for R, and so on. In a polyalphabetic cipher, any letter may replace the message letter. This makes any attempt to break the cipher very difficult.

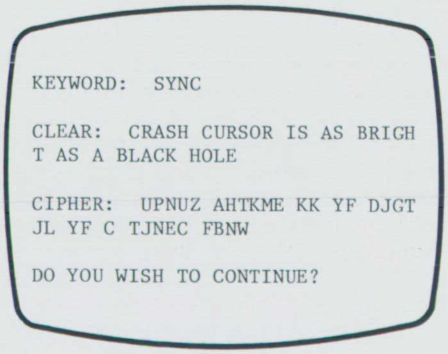
James John Hollandsworth, Box 163, Montcoal, WV 25135.

Figure 1. The Vigenere Tableau

Plain-text letter

Key letter	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 2. Sample Run



```
KEYWORD:  SYNC

CLEAR:  CRASH CURSOR IS AS BRIGH
T AS A BLACK HOLE

CIPHER:  UPNUZ AHTKME KK YF DJGT
JL YF C TJNEC FBNW

DO YOU WISH TO CONTINUE?
```

The basis of the Vigenere cipher is a table known as the Vigenere tableau (what else?) shown in Figure 1. Although there are several versions of the cipher, this article deals with only two. The first uses a *keyword* known to both the sender and receiver. In the program the prompt will ask you to INPUT KEYWORD (1-12 LETTERS). Naturally, you may change the keyword whenever and as often as you wish as long as your receiver knows the change. To demonstrate the program, let us suppose that you must for some obscure reason send this message to a fellow ZX80 user: CRASH CURSOR IS AS BRIGHT AS A BLACK HOLE. When the screen calls for the keyword, enter SYNC. The display will prompt: INPUT TEXT. On 1K RAM the number of letters and spaces (no punctuation marks) you can enter without overloading the memory is about 52. Of course, you can do longer messages by breaking them up into units of fewer than 52 characters. You will then be asked: IS THIS A CLEAR OR A CIPHER? Clear is the technical term for your message. Type it in. In a few seconds the screen will appear as in Figure 2. Copy down the cipher, answer NO to get out of the program, or YES if you have some more messages. Send the message to your friend who will follow the same routine, except that he types in CIPHER when asked, and in a few seconds he will read your message.

Listing 1. Vigenere Cipher Program

Keyword Form (4K ROM; 1K RAM)

```
   1 PRINT "VIGENERE CIPHER"
   2 PRINT
   3 PRINT
   4 DIM A(12)
   5 PRINT "INPUT KEYWORD (1-12 L
ETTERS)"
  10 INPUT A$
  11 CLS
  12 LET E$=A$
  20 FOR Y=1 TO 12
  30 LET A(Y)=CODE(A$)-38
  40 LET A$=TL$(A$)
  45 IF CODE(A$)=1 THEN GO TO 55
  50 NEXT Y
  55 PRINT "INPUT TEXT"
  60 INPUT C$
  61 CLS
  70 LET T=1
  71 PRINT "IS THIS A CLEAR OR A#
CIPHER?"
  72 INPUT Z$
  73 CLS
  74 PRINT "KEYWORD:##";E$
  75 PRINT
  76 PRINT Z$;":"
  77 PRINT C$
  78 PRINT
  79 IF Z$="CIPHER" THEN GO TO 16
0
  80 PRINT "CIPHER:"
  83 LET B=CODE(C$)+A(T)
  85 IF CODE(C$)=0 THEN LET B=0
  90 IF B>63 THEN LET B=(B-63)+37
```

```
100 PRINT CHR$(B);
110 IF NOT B=0 THEN LET T=T+1
120 IF T>Y THEN LET T=1
130 LET C$=TL$(C$)
140 IF NOT CODE(C$)=1 THEN GO TO
83
150 GO TO 230
160 PRINT "CLEAR:"
165 LET B=CODE(C$)-A(T)
167 IF CODE(C$)=0 THEN LET B=0
170 IF B<38 AND B>0 THEN LET B=6
4-(38-B)
180 PRINT CHR$(B);
190 IF NOT B=0 THEN LET T=T+1
200 IF T>Y THEN LET T=1
210 LET C$=TL$(C$)
220 IF NOT CODE(C$)=1 THEN GO TO
165
230 PRINT
235 PRINT
240 PRINT "DO YOU WISH TO CONTIN
UE?"
260 INPUT W
270 CLS
280 RUN
```

Notes:

4 Array for character codes of keyword.

5 Underline indicates use of word from key board if desired to save memory.

10 Keyword.

12 Saves keyword for later display.

20 Loop to load keyword character codes into array.

45 Takes program out of loop for keywords of fewer than 12 letters.

70 Initializes a value used in coding/decoding routines.

72 Nature of text.

74 Displays keyword.

76 Displays, identifies text.

79 Check if coding or decoding routine is to be used.

83 Steps up letter according to value in array.

85 Checks for space in message.

90 Loops around if coded letter Z.

100 Prints coded letter.

110 Increments keyword letter array unless space.

120 Loops around keyword if end is reached.

130 Next letter.

140 Checks for end of text.

160 to 210 Decoding routine is essentially the reverse of the coding routine.

The other type of Vigenere cipher we will consider is based on one letter and is a variation of the autokey system, so called because the key letter changes automatically in the message. To understand this cipher better, let us manually encipher the above message. To make the cipher even harder to break, divide the message into five-letter groups (this can be done with the keyword type also): CRASH CURSO RISAS BRIGH TASAB LACKH OLE. To encipher this you must first pick a key letter both you and your receiver have agreed upon in advance. Picking S as the

The other type of the Vigenere cipher we will consider is based on one letter and is a variation of the autokey system, so called because the key letter changes automatically in the message. To understand this cipher better, let us manually encipher the above message. To make the cipher even harder to break, divide the message into five-letter groups (this can be done with the keyword type also): CRASH CURSO RISAS BRIGH TASAB LACKH OLE. To encipher this you must first pick a key letter both you and your receiver have agreed upon in advance. Picking S as the

key letter, we take the first letter C. Look down the plain-text column in Figure 1 to where it meets the S key letter row. At the intersection is U. This is the first letter of your message. Repeat this process with each letter and eventually you will have: ULLDK MGXPD UCUUM NEMSZ SSKKL WWYIP DOS. Since the key letter and the cipher itself changes throughout the message, it is both extremely difficult to break and extremely difficult to encipher and to decipher. However, the ZX80 removes that difficulty. Make the modifications in the program in Listing 1 as shown in Listing 2. The procedure on the computer is the same as for the keyword except that you will INPUT a *key letter*. With the changes you will have about 90 letters and spaces for one message in 1K RAM.

Listing 2. Vigenere Cipher Program

Modification for Key Letter Form

```
DELETE Lines 4, 12, 30, 40, 45,
  50, 70, 120, 200.

Make the following line changes:

  5 PRINT "INPUT KEY LETTER"
 20 LET Y=CODE(A$)-38
 74 PRINT "KEY LETTER:##";A$
 83 LET B=CODE(C$)+Y
110 IF NOT CODE(C$)=0 THEN LET Y
=B-38
165 LET B=CODE(C$)-Y
190 IF NOT CODE(C$)=0 THEN LET Y
=CODE(C$)-38
```

Now let us see how the program works. If you closely study the tableau, you will see that the key letter A row does not really change the message. The key letter B row actually moves the message letter up one—message letter A becomes cipher letter B, B becomes C, etc. The key letter C row moves the message letter up two. The message letter then is moved the number of letters the key letter is beyond A. Also notice that when you have to move a letter so that it is beyond Z, the table wraps around back to A. So to program this I made use of the ZX80's character code functions that allow it to treat letters as numbers as the basis of the program. When the key letter is INPUTed, its character code -38 is stored in Y (see Listing 2). This gives the number a range of A=0 to Z=25. To encipher, the program takes the code of the first letter and adds it to the value of Y. If the resulting sum is greater than 63 (the character code for Z), the program wraps around, adding the value beyond 63 to 37. It then prints the character of the manipulated variable B and goes on the next letter, exchanging the present value of Y for the value of B-38. Deciphering is accomplished in a similar manner. The program duplicates the functioning of the cipher table.

For further information on the Vigenere cipher and other ciphers and codes, I would suggest checking:

Codes, Ciphers, and Secret Writing by Martin Gardner (a good beginner's book).

The Codebreakers by David Kahn (a mammoth book describing in great detail the entire history of cryptology and cryptologists).

And, of course, encyclopedia articles on codes and ciphers provide a quick introduction.

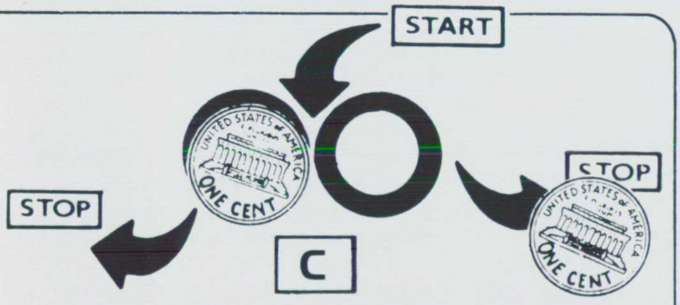
8K Version

The program can be converted to 8K by making the changes in Listing 3. However, for 8K you will need additional memory.

Listing 3.Vigenere Cipher Program Modifications

Key Letter Form (8K ROM; over 1K RAM)

```
  1 PRINT "VIGENERE CIPHER"
  2 PRINT
  3 PRINT
  5 PRINT " INPUT KEY LETTER"
 10 INPUT A$
 11 CLS
 20 LET Y=CODE A$-38
 55 PRINT " INPUT TEXT"
 60 INPUT C$
 61 CLS
 71 PRINT "IS THIS A CLEAR OR C
IPHER?"
 72 INPUT Z$
 73 CLS
 74 PRINT "KEY LETTER:   ";A$
 75 PRINT
 76 PRINT Z$;":"
 77 PRINT C$
 78 PRINT
 79 IF Z$="CIPHER" THEN GOTO 16
0
 80 PRINT "CIPHER:"
 83 LET B=CODE C$+Y
 85 IF CODE C$=0 THEN LET B=0
 90 IF B>63 THEN LET B=(B-63)+3
7
100 PRINT CHR$ B;
110 IF NOT CODE C$=0 THEN LET Y
=B-38
130 LET C$=C$(2 TO )
140 IF NOT C$="" THEN GOTO 83
150 GOTO 230
160 PRINT " CLEAR :"
165 LET B=CODE C$-Y
167 IF CODE C$=0 THEN LET B=0
170 IF B<38 AND B>0 THEN LET B=
64-(38-B)
180 PRINT CHR$ B;
190 IF NOT CODE C$=0 THEN LET Y
=CODE C$-38
210 LET C$=C$(2 TO )
220 IF NOT C$="" THEN GOTO 165
230 PRINT
235 PRINT
240 PRINT "DO YOU WISH TO CONTI
NUE?"
260 INPUT W
270 CLS
280 RUN
```



The Root of All Evil

Playing with money can get you into trouble, all right. But it can also teach you and your children what happens inside a computer.

By sliding and flipping pennies (affluent readers can use dimes) you learn exactly how simple computer circuits work.

The first half of *Computer Coin Games* provides directions and diagrams for a variety of games which can be played by anyone—computer enthusiast or not. The second half of the book explains how the games relate to computers.

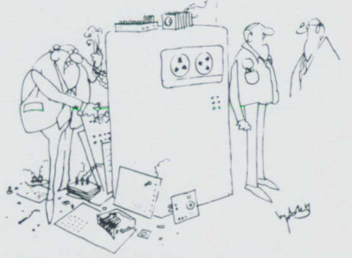
Computer Coin Games is an inexpensive, entertaining way to introduce children and adults to binary numbers, flip flops and counters. Order your copy today.

To order send $3.95* plus $3.00 postage and handling to Creative Computing, Dept. Z0125, One Park Avenue, Room 458, New York, N.Y. 10016. Orders may also be charged to your Visa, MasterCard or American Express account—by mail include credit card name, number and expiration date, or if you prefer, use our 24 hour toll-free number, 800-631-8112. In N.J. only 201-540-0445.

*Residents of CA, CO, DC, FL, IL, MA, MI, MO, NJ, NY State, OH, SC, TN, and VT add applicable sales tax.

creative computing

Morris Plains, NJ 07950
Toll-free **800-631-8112**
(In NJ 201-540-0445)



The Best Medicine

"He said he could fix it in half the time it was taking me, so I let him try..."

Your program is finally finished. You type RUN. Your computer replies READY. It has just eaten five hours of your life. What do you do?

Try to maintain your sense of humor—reach for *The Colossal Computer Cartoon Book*. Laugh at the original adventures of Edu-Man. Chuckle at cartoons by *Creative Computing* favorites Sandy Dean, Harbaugh, Swan, and Johns. Smile sympathetically at the ways others have suggested to get even with the infernal machines.

The Colossal Computer Cartoon Book contains hundreds of clever cartoons to tickle the funny bone of anyone who has ever loved or hated a computer.

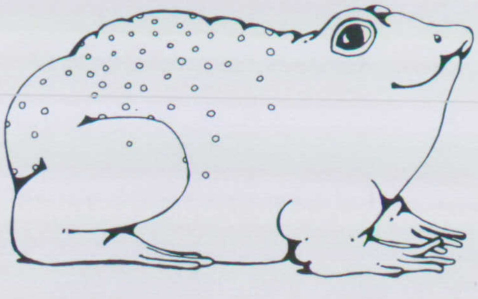
To order send $4.95* plus $3.00 postage and handling to Creative Computing, Dept. 20124, One Park Avenue, Room 458, New York, NY 10016. Orders may also be charged to your Visa, MasterCard or American Express account—by mail include credit card name, number and expiration date, or if you prefer, use our 24 hour toll-free number, 800-631-8112. In NJ only 201-540-0445.

*Residents of CA, CO, DC, FL, IL, MA, MI, MO, NJ, NY State, OH, SC, TN, and VT add applicable sales tax.

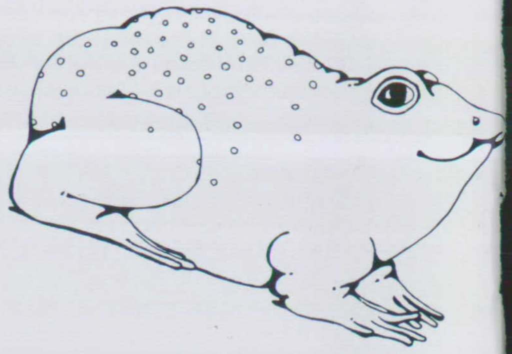
creative computing

Morris Plains, NJ 07950
Toll-free **800-631-8112**
(In NJ 201-540-0445)

4K ROM
2K(1K) RAM



Tioga Toads



Jerry Ginn

Inverse characters, including graphic symbols can be included in program lines as either literals or string variables, resulting in improved displays and faster graphics. Titles, headings, and prompts are examples of display items that can benefit from the added emphasis of inverse printing. Full use of the ZX80 character set also requires inverse characters.

The ZX80 achieves printing by accessing a separate set of codes in the character set that is contained in the ROM. Normal characters are assigned codes from 0 to 63 and inverse characters have codes from 128 to 191. Code values from 64 to 127 and from 192 to 211 have no valid interpretation and their use with the CHR$ function will print a question mark ("?"). Codes above 212 are reserved for Basic tokens.

You can use the CHR$(n) with any valid code to print a character. The program statement 10 PRINT CHR$(38) will print the normal character "A", and the line 10 PRINT CHR$(166) will print the inverse character " $\boxed{\text{A}}$ ". The first line is equivalent to 10 PRINT "A", which takes fewer bytes and executes faster, but, because the keyboard has no $\boxed{\text{A}}$ key, there is no direct equivalent for the second statement.

The limitation is in the input, not the interpretation. If the program contained the line 10 PRINT "A", it would be properly interpreted and the result would be an inverse " $\boxed{\text{A}}$ " on the display.

You can construct that line by typing NEW and entering these lines:

```
10 PRINT "A"
20 POKE 16428,166
```

RUN this and a normal "A" will appear on the screen. When you return to the listing, you will find that the first line has been altered to read:

10 PRINT " $\boxed{\text{A}}$ "

This has happened because Basic programs begin at location 16424. The first two bytes contain the line number; byte three holds the decimal value 244 which is the code value for PRINT; byte four has the value 1, code for a quotation mark and byte five (address 16428) has the value 38, the code for "A". Line 20 sets this byte equal to 166 (erasing 38). Once you have RUN this, line 20 is no longer needed and can be deleted.

If you want to use the " $\boxed{\text{A}}$ " as a string variable, edit line 10. Type SHIFT/NEWLINE and then SHIFT/8 to remove the cursor to the right of PRINT. Type SHIFT/0 to delete PRINT and then type LET A$ = and NEWLINE. Your program should now show:

10 LET A$=" $\boxed{\text{A}}$ "

You can also use the edit mode to relocate the line anywhere in your program by changing the line number.

The effect is even more valuable when applied to longer strings of characters. Try this:

```
 10 PRINT "XXXXXXXXX" (9 X's)
 20 POKE 16428,128
 30 POKE 16429,173
 40 POKE 16430,174
 50 POKE 16431,185
 60 POKE 16432,128
 70 POKE 16433,179
 80 POKE 16434,149
 90 POKE 16435,177
100 POKE 16436,128
```

When you RUN this you will have created the line:

10 PRINT " $\boxed{\text{HIT N/L}}$ "

This line now occupies 14 bytes. To achieve this result using CHR$ would require a total of 93 bytes, almost seven times as many bytes. Note that the string of characters begins and ends with an inverse space to improve readibility. Here is a prompt with emphasis!

The method shown here can be used with any valid character code but some caution is required. When POKE is used to set the value of a byte, the previous value of that byte is lost. If that byte held code that was critical to the interpretation of the statement, then a syntax error or a crash could result. In the example above, the addresses 16427 and 16437 contain the string delimiting quotes. To replace these with any other value would cause a syntax error. Address 16438 contains the NEWLINE character code 118 that ends line 10. To replace it would cause a crash. It is therefore necessary that you have an "X" or other dummy character or space in line 10 for each character or space that you want in the final version of the string. A crash may also occur if you POKE an invalid character code or certain Basic token codes into a program line and list the line on the screen. So stick to the values 0 through 63, and 128 through 191. The code value 1 can also cause you difficulty. Because it is the sting delimiter, it will be interpreted as the end of the string, not as the quotation mark or ditto character. The codes for the ditto that can be used within a string are 129 for inverse and 212 for normal.

Jerry Ginn, P. O. Box 30, Shawville, PA. 16873.

Entering all of these POKE's and addresses can be a pain in the neck so I use a short routine which provides an onscreen progress review, allows for corrections and checks for the end of dummy string.

```
 1 LET AD=16440
 2 PRINT "ANY DUMMY STRING"
10 IF PEEK(AD)=1 THEN LIST 2
11 INPUT C
12 IF C<0 THEN LET AD=AD-1
13 POKE AD,ABS(C)
14 CLS
15 LET AD=AD+1
16 GO TO 2
```

I use this routine during program entry to build any necessary inverse strings or graphic strings and then delete it from the final version. The routine displays the dummy string, and, as you enter each code, the change is made visible on the screen. If you entered the wrong code, you can correct the last character entered by entering the correct code as a negative.

When all of the codes have been entered, the display returns to the listing with the cursor at line 2 for editing. The dummy string statement can be numbered 2 through 9 as long as it is the second line of the program. PRINT statements numbered 2 through 9 will be printed on the screen, but only the second program line will be modified. This allows you to build multi-line graphic statements and check the whole display before editing.

To construct the toad in the program that follows, use a 3 character dummy to create each of these lines:

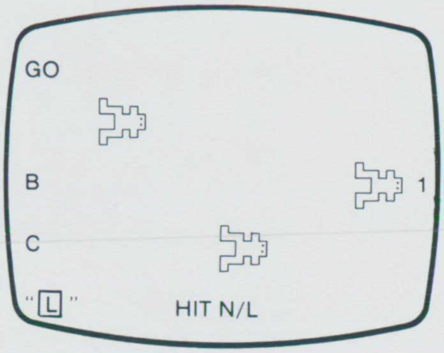
2 PRINT "▘▜▜"	(133,6,6)
3 PRINT "█□⁚"	(0,128,142)
4 PRINT "▗▙▙"	(135,4,4)

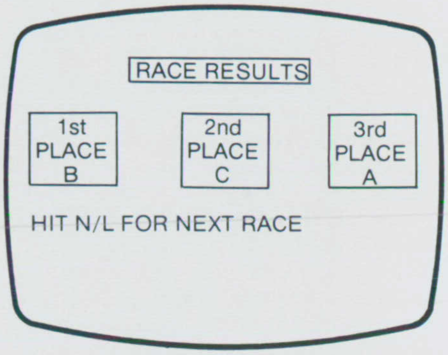
Start with line 4, entering the codes for the bottom third of the toad. Leave line 4 where it is and enter the dummy for line 3. When it is complete, leave it in place and enter the dummy for line 2. Each line will be modified in turn and added to the top of the previous construction. Having the the entire toad on screen will help to spot errors. After all three lines are created, they can be edited to locate them at the proper location in the program by changing line number 2 to 960, 3 to 980, and 4 to 1000.

In the following program, three of these toads hop across the screen and announce their order of arrival at the finish line. When all of the toads have finished, the race results are printed on a scoreboard making lavish use of inverse characters. If you have 2K or more of RAM, the progam can be expanded to allow two players to enter their opinion as to which toad will win the race and to comment on each player's choice after the race.

The 1K version runs within 6 bytes of the memory limit. Using CHR$ to print all of the inverse character would require more than 400 additional bytes and would exceed the limit.

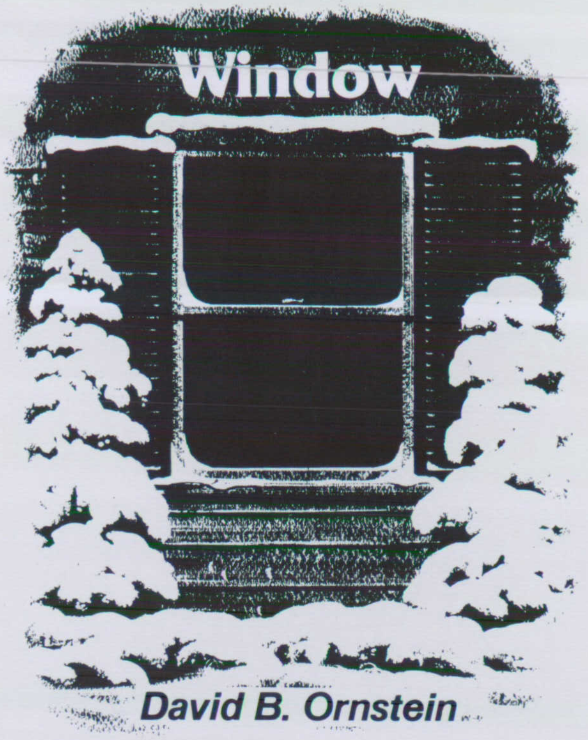
The 2K version requires just over 1400 bytes to run, leaving about 600 bytes for additional expansion such as more toads, more players, bets on the outcome or the use of machine language routines such as found in Dr. Logan's "Auto Display Changing" article (*SYNC* 1:3). ▚





```
**    1 LET AD=16440
**    2 PRINT "ANY STRING"
**   10 IF PEEK(AD)=1 THEN LIST 2
**   11 INPUT C
**   12 IF C<0 THEN LET AD=AD-1
**   13 POKE AD,ABS(C)
**   14 CLS
**   15 LET AD=AD+1
**   16 GO TO 2

*   100 DIM T(1)
    110 DIM P(2)
    120 DIM F(2)
    130 DIM R(2)
    140 LET A$="GET READY"
*   150 CLS
    160 LET F$="123"
*   180 PRINT
*   190 PRINT, "TIOGA COUNTY"
*   200 PRINT, " TOAD RACES "
*   210 PRINT, "            "
*   230 PRINT
*   240 PRINT
*   250 PRINT "PICK YOUR TOAD...(A,B
        OR C)"
*   260 PRINT
*   270 FOR N=0 TO 1
*   280 PRINT "PLAYER ";N+1;"? ";
*   290 INPUT U$
*   300 PRINT U$
*   310 LET T(N)=CODE(U$)-38
*   320 PRINT
*   330 NEXT N
*   340 PRINT
*   350 PRINT "HIT N/L TO START RACE"
*   360 PRINT
*   370 INPUT U$
    390 CLS
    400 PRINT
    410 PRINT A$
    420 PRINT
    430 FOR T=0 TO 2
    440 PRINT
    450 FOR L=1 TO 3
    460 PRINT CHR$((T+38)*ABS(L=2));
    470 IF P(T)=0 THEN GO TO 510
    480 FOR N±1 TO P(T)
    490 PRINT "####";
    500 NEXT N
    510 GO SUB L*20+940
    520 NEXT L
    530 PRINT
    540 NEXT T
    550 PRINT ,"HIT N/L"
    560 INPUT U$
    570 CLS
    575 LET A$="GO"
    580 IF F$="" THEN GO TO 700
    590 LET X=RND(3)-1
    600 IF P(X)=5 THEN GO TO 590
    610 LET P(X)=P(X)+1
    620 IF P(X)=5 THEN GO SUB 640
    630 GO TO 390
    640 LET F(X)=CODE(F$)
    650 LET F$=TL$(F$)
    660 LET R(F(X)-29)=X
    670 RETURN
    700 PRINT
    710 PRINT
    720 PRINT,"RACE RESULTS "
    730 PRINT
    740 PRINT
    750 PRINT " 1ST ","  2ND ","  3RD "
    760 PRINT "PLACE","PLACE","PLACE"
    770 FOR N=0 TO 2
    780 PRINT "□□";CHR$(R(N)+166);"□□",
    790 NEXT N
*   800 FOR N=0 TO 1
*   810 PRINT
*   820 PRINT "PLAYER ":N+1:
*   830 GO SUB (F(T(N))-29)*20+900
*   840 NEXT N
    850 PRINT
    860 PRINT "HIT N/L FOR NEXT RACE"
    870 PRINT
    880 INPUT U$
    890 RUN
*   900 PRINT " IS A FINE JUDGE"
*   905 PRINT ,"OF TOAD FLESH."
*   910 RETURN
*   920 PRINT " JUST MISSED."
*   930 RETURN
*   940 PRINT " IS A TURKEY."
*   950 RETURN
    960 PRINT "▘▜▜"
    970 RETURN
    980 PRINT "█□⁚"
    990 RETURN
   1000 PRINT "▗▙▙"
   1010 RETURN

NOTE: LINES MARKED ** CAN BE ERASED AFTER ALL INVERSE
      STRINGS AND GRAPHICS ARE CREATED.
      LINES MARKED * MUST BE DELETED TO RUN IN 1K
```

8K ROM
16K RAM



Window

David B. Ornstein

Since the ZX81 uses the powerful Z80 microprocessor, it is a good system for which to write machine code programs. In the past months while writing many programs, I have found that there is one feature lacking on the system: a machine language monitor.

A machine language monitor is a utility, provided by most computer systems, which aids in the development of machine language programs. Its basic functions are: a) to allow you to view the contents of each byte in the system's memory, and b) to allow you to change these values.

The program provided here will allow you to perform these functions. It is a visual window into the system's memory, hence the name. In addition, it is a program which illustrates the programming litany: a program should contain very few constants intermixed with its code.

Looking at the program, you will see that the first few score of lines are all assignment statements (LETs). All arbitrary constants are specified in this section of the program. All references to these values later in the program are, then, symbolic, making the code easier to read.

Another benefit of coding the program this way is the ease of modification it provides. For example, you can change the line on the screen on which the 'window' begins by modifying the value of the variable PRITOP (PRImary screen TOP). Most of the visual arrangement can be changed by changing the value in one or two LET statements.

David B. Ornstein, 25 Shute Path, Newton, MA 02159.

The commands for WINDOW are:

- K- The K key (+) is used to move to the next memory location.
- J- The J key (-) is used to move to the previous memory location.
- G- The G key (GOTO) is used to move the current location to wherever you choose. The system will ask you for an address in hex.
- P- The P key (PRINT) is used to list out the contents of 10 memory locations, in hex and as characters. The PRINT starts at the current location and, when done, sets the current location equal to the next location.
- D- The D key (DISPLAY) is used to print out 5 lines of characters which are the characters in memory, from the current location on.
- Q- The Q key (QUIT) is used to exit the WINDOW program. It will leave you in FAST mode. You can re-enter the program with CONTinue.
- C- The C key (CALL) is used to call a machine language routine. Its address is specified by the current location address.
- O- The O key (OPEN) is used to change the contents of the current location. The system will prompt you for a 2-digit hex value. The system will then increment the current location pointer. ▚

Ed.—For those who do not want to do the work of entering the program, but who do want to enjoy its benefits, it can be obtained on cassette from Heuristics, 25 Shute Path, Newton, MA 02159 for $8.00.)

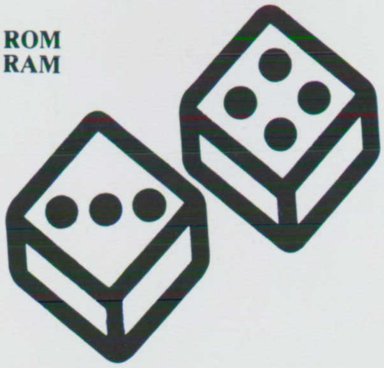
```
   1 REM 123456789012345678901..
.........
   2 REM        WINDOW

        COPYRIGHT (C), 1981
        BY:
            HEURISTICS

        ANY COPY OF THIS
        PROGRAM, PRINTED
        OR MACHINE-READABLE,
        MUST INCLUDE THIS
        COPYRIGHT NOTICE.

   5 SLOW
  10 LET HEX16=2000
  12 LET DISP=1000
  14 LET SPLITBYTE=2100
  16 LET INVERSE=2200
  18 LET HEX8=2300
  20 LET GETKEY=2400
  22 LET C$="GKJDQPCO"
  24 LET INPUT=2500
  26 LET DEC16=2600
  28 LET AD=16514
  30 LET VALX=15
  32 LET CLRSEC=2700
  34 LET DEC8=2800
  36 LET CLRPMT=2900
  40 LET LPD=22
  42 LET PMTY=LPD-1
  44 LET PRITOP=1
  46 LET PRIBOT=5
  48 LET SECTOP=PRIBOT+1
  50 LET SECBOT=PMTY-1
  52 LET SECSIZ=SECBOT-SECTOP+1
  54 LET NUM=28
  56 LET FLY=PRITOP+(PRIBOT-PRIT
OP)/2
```

```
  58 LET FLV=6
  60 LET INVERT=128
  62 LET ADX=7
  64 LET MAXADDR=65535
  66 LET MINADDR=0
  68 LET FLX=30
  70 LET CURY=16442
  72 LET LPP=SECSIZ
  74 LET CHARX=20
  76 LET DL=SECSIZ
  78 LET MAIN=100
  80 LET MAIN2=130
  82 LET DISP2=1040
  84 LET DISP3=1080
 100 REM MAIN
 120 GOSUB DISP
 130 GOSUB GETKEY
 132 PRINT AT FLY,FLX;"█"
 135 IF Z=5 THEN GOTO 150
 137 IF Z>5 THEN LET Z=Z-1
 140 GOTO 3000+(Z-1)*1000
 150 FAST
 155 STOP
 157 SLOW
 160 GOTO MAIN
1000 REM DISPLAY
1010 FOR V=PRITOP TO PRIBOT
1020 PRINT AT V,0;"████████████
████████████████████"
1030 NEXT V
1040 PRINT AT FLY,1;"ADDR█";
1045 LET Z=AD
1050 GOSUB HEX16
1070 PRINT AT FLY,ADX;Z$;
1080 PRINT AT FLY,VALX;
1090 LET Z1=PEEK AD
1100 GOSUB HEX8
1110 PRINT Z$
1120 RETURN
2000 REM HEX16
```

```
2010 LET Z1=INT (Z/256)
2020 LET Z2=Z-Z1*256
2030 GOSUB SPLITBYTE
2040 LET Z$=CHR$ (Z8+NUM)+CHR$ (
Z9+NUM)
2050 LET Z1=Z2
2060 GOSUB SPLITBYTE
2070 LET Z$=Z$+CHR$ (Z8+NUM)+CHR
$ (Z9+NUM)
2080 RETURN
2100 REM SPLITBYTE
2110 LET Z8=INT (Z1/16)
2120 LET Z9=Z1-Z8*16
2130 RETURN
2200 REM INVERSE
2210 FOR Z=1 TO LEN Z$
2220 LET Z$(Z)=CHR$ (CODE Z$(Z)+
INVERT)
2230 NEXT Z
2240 RETURN
2300 REM HEX8
2310 GOSUB SPLITBYTE
2320 LET Z$=CHR$ (Z8+NUM)+CHR$ (
Z9+NUM)
2330 RETURN
2400 REM GETKEY
2405 LET CB=FLV
2410 LET Z$=INKEY$
2415 PRINT AT FLY,FLX;CHR$ CB;
2417 GOSUB 2470
2420 IF Z$="" THEN GOTO 2410
2430 FOR Z=1 TO LEN C$
2440 IF Z$=C$(Z) THEN RETURN
2450 NEXT Z
2460 GOTO 2410
2470 IF CB=FLV+INVERT THEN GOTO
2490
2475 LET CB=CB+INVERT
2480 RETURN
2490 LET CB=CB-INVERT
```

```
2495 RETURN
2500 REM INPUT
2510 INPUT Z$
2520 RETURN
2500 REM DEC16
2610 LET Z$="0000"(1 TO 4-LEN Z$
)+Z$
2620 LET Z=0
2630 FOR G=1 TO 4
2640 LET Z=Z*16
2650 LET Z=Z+CODE Z$(G)-NUM
2660 NEXT G
2670 RETURN
2700 REM CLRSEC
2710 FOR B=SECTOP TO SECBOT+1
2720 PRINT AT B,0;"
                               "

2730 NEXT B
2740 PRINT AT SECTOP,0;
2750 RETURN
2800 LET Z$="00"(1 TO 2-LEN Z$)+
Z$
2805 LET Z=16*CODE Z$+CODE Z$(2)
-476

2810 LET Z=(CODE Z$-NUM)*16+CODE
 Z$(2)-NUM
2820 RETURN
2900 REM CLRPMT
2905 PRINT AT PMTY,0;"
                                ";

2910 RETURN
3000 REM JUMP
3005 GOSUB 3200
3010 PRINT AT FLY+1,ADX;"    "
3015 PRINT AT PMTY,0;"ENTER THE
ADDRESS IN HEX."
3020 GOSUB INPUT
3025 GOSUB CLRPMT
3030 IF LEN Z$>4 THEN GOTO 3020
3040 GOSUB DEC16
3060 PRINT AT FLY+1,ADX;Z$
3070 FOR V=ADX TO ADX+3
3080 PRINT AT FLY+1,V;"█"
3090 PRINT AT FLY,V;Z$(V-ADX+1)
3100 NEXT V
3120 LET AD=Z
3130 GOSUB DISP3
3140 GOTO MAIN2
3200 LET Z=AD
3210 GOSUB HEX16
3220 GOSUB INVERSE
3230 PRINT AT FLY,ADX;Z$
3240 RETURN
4000 REM INC ADDR
4010 LET AD=AD+1
4020 IF AD>MAXADDR THEN LET AD=M
INADDR
4025 GOSUB 1040
4030 GOTO MAIN2
5000 REM DEC ADDR
5010 LET AD=AD-1
5020 IF AD<MINADDR THEN LET AD=M
AXADDR
5025 GOSUB DISP2
5030 GOTO MAIN2
6000 REM DISPLAY
6001 LET TC=AD
6003 GOSUB CLRSEC
6005 PRINT AT SECTOP,0;
6010 LET IL=LPD-PEEK CURY
6030 IF LPD-PEEK CURY>=DL+IL OR
INKEY$<>"" THEN GOTO MAIN2
6033 PRINT CHR$ (PEEK TC);
6035 LET TC=TC+1
6040 GOTO 6020
7000 REM PRINT
7005 GOSUB CLRSEC
7010 FOR P=0 TO LPP-1
7015 IF INKEY$<>"" THEN GOTO 130
7020 LET Z=AD+P
7030 GOSUB HEX16
7040 PRINT TAB ADX;Z$;TAB VALX;
7050 LET Z1=PEEK Z
7055 GOSUB HEX8
7060 PRINT Z$;TAB CHARX;CHR$ (Z1
)

7070 NEXT P
7080 LET AD=AD+P
7090 GOTO MAIN
8000 REM CALL
8005 GOSUB CLRSEC
8010 LET KK=USR AD
8020 PRINT AT SECTOP,0;"THE CALL
 SET BC=";
8030 LET Z=KK
8040 GOSUB HEX16
8050 PRINT Z$
8060 GOTO MAIN
9000 REM OPEN
9020 LET Z1=PEEK AD
9030 GOSUB HEX8
9040 GOSUB INVERSE
9060 FOR N=FLY+1 TO PRIBOT-1
9065 PRINT AT N,VALX;"  "
9070 PRINT AT N-1,VALX;Z$
9080 LET Z$="██"
9095 NEXT N
9097 PRINT AT PMTY,0;"ENTER THE
VALUE IN HEX."
9100 GOSUB INPUT
9110 GOSUB CLRPMT
9120 IF LEN Z$>2 THEN GOTO 9100
9130 GOSUB DEC8
9140 PRINT AT PRIBOT-1,VALX;Z$
9150 FOR N=PRIBOT-1 TO FLY+1 STE
P -1
9160 PRINT AT N,VALX;"██"
9170 PRINT AT N-1,VALX;Z$
9180 NEXT N
9190 POKE AD,Z
9195 LET AD=AD+1
9200 GOTO MAIN
```

8K ROM
1K RAM



Dice and Train

Joseph R. Sutton

I recently got a copy of Creative Computing's *Basic Computer Games* and of *More Basic Computer Games* to find more challenge for my 8K ROM and 16K RAM ZX80, but before turning to the really long programs I could not resist trying to squeeze two of the smaller ones into 1K for the 8K ROM. The results are given below. In both cases the screen prompts will ask for INPUT which should be followed by NEWLINE.

Dice strictly speaking is not a game. By simulating the throw of a pair of dice, the computer calculates the distribution of the number of throws entered according to the number of spots the dice show. Be prepared for a wait if you enter a number like 5000.

Train likewise is not exactly a game, but rather a challenge to balance time, speed, and distance factors. You have a chance to do some mental calculation and then have the computer tell you how close you came to being right. ▚

Joseph R. Sutton, 170 S. Hillside Ave., Succasunna, NJ 07876.

Listing 1. Dice

```
  5 RAND
 10 DIM F(12)
 80 FOR Q=1 TO 12
 90 LET F(Q)=0
100 NEXT Q
105 CLS
110 PRINT "HOW MANY ROLLS?"
120 INPUT X
125 PRINT X
130 FOR S=1 TO X
140 LET R=INT (6*RND+1)+INT (6*
RND+1)
170 LET F(R)=F(R)+1
180 NEXT S
190 PRINT "TOTAL SPOTS","NO. OF
TIMES"
200 FOR V=2 TO 12
210 PRINT V,F(V)
220 NEXT V
222 PRINT "TRY AGAIN?"
223 INPUT Z$
224 IF Z$="YES" THEN GOTO 80
```

Sample Run

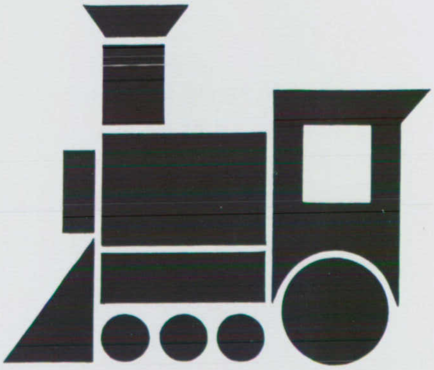
```
HOW MANY ROLLS?
100
TOTAL SPOTS       NO. OF TIMES
2                 1
3                 8
4                 12
5                 11
6                 11
7                 11
8                 14
9                 10
10                8
11                10
12                4
TRY AGAIN?
```

Listing 2. Train

```
  5 CLS
 10 LET C=INT (25*RND)+40
 15 LET D=INT (15*RND)+5
 20 LET T=INT (19*RND)+20
 25 PRINT AT 6,2;"A CAR TRAVELI
NG ";C;" MPH CAN","MAKE A TRIP I
N ";D;" HR LESS THAN"
 30 PRINT "A TRAIN TRAVELING AT
 ";T;" MPH."
 35 PRINT " HOW LONG WOULD IT T
AKE BY CAR?"
 40 INPUT A
 45 LET V=D*T/(C-T)
 50 LET E=INT (ABS ((V-A)*100/A
)+.5)
 55 IF E>5 THEN GOTO 70
 60 PRINT "GOOD ANSWER WITHIN "
;E;" PERCENT."
 65 GOTO 80
 70 PRINT "SORRY. OFF BY ";E;"
PERCENT"
 80 PRINT "ANSWER IS ";V;" HR."
 85 PRINT "AGAIN? (Y OR N)"
 90 INPUT A$
 95 IF A$="Y"  THEN RUN
```

Sample Run

```
  A CAR TRAVELING 41 MPH CAN
MAKE A TRIP IN 14 HR LESS THAN
A TRAIN TRAVELING AT 37 MPH.
 HOW LONG WOULD IT TAKE BY CAR?
GOOD ANSWER WITHIN 2 PERCENT.
ANSWER IS 129.5 HR.
AGAIN? (Y OR N)

(INPUT YOUR GUESS: 120)
```



resources

Users Groups

•DATAmerica Computer Users Group. Membership is free; publishes the *DATAmerica Newletter* for the cost of postage to your location. Exchange programs, ideas, etc. For Hardware and Software oriented people and most personal computer systems. Send U.S. postage stamp or 20 cents (International subscribers send U.S. funds sufficient for postage to their nation) to:

DATAmerica Computer Users Group
312 E. 84 St., #1A
New York, NY 10028

•C.A.C.H.E., Inc.
(Chicago Area Computer Hobbyist Exchange): 700 members; monthly newsletter; monthly meetings at DeVry Institute of Technology, Chicago Campus, 3rd Sunday of the month between ll a.m. and 4 p.m.; membership: $10 annually.

Sinclair ZX S.I.G. (Special Interest Group)
Send SASE for membership application to:

C.A.C.H.E., Inc.
Z80/Sinclair ZX S.I.G.
L. P. Weigel, Coordinator
Box C-176
323 S. Franklin, #804
Chicago, IL 60606

•Club Nacional de Usuarios del ZX81
A Spanish users club founded in November 1981 with over 200 members. Bulletin issued 3 times annually; membership: 1.200,-ptas includes 4 consecutive issues of the bulletin. For information write:

Club Nacional de Usuarios del ZX81
Avda. de Madrid
no. 203-207, 1o, 3a, esc. A
Barcelona, 14
Spain

RAM Expansion

•Add on RAM for ZX80; 2K increments up to 16K. Complete schematic, parts list, sources, and how-to for $3.95. Appr. cost for 2K, $20; ea. additional, the cost of the chip; no additional power supply needed. Send $3.95 to:

Dennis Weber
PO Box 742
Troutman, NC 28166

• For additional 1K RAM, plug in (as described in *Syntax ZX80*, order a Factory Prime HIT 6116-3 16K (2Kx8) CMOS RAM (150nS) for $14.50. Also available for 8K expansion, 2114L-2 (1Kx4) Static Rams (200nS) for $3.25 each or set of 16 for $48. $1.25 for shipping and postage. U.S. funds only. Order from:

Steven Streebing
Box 182
Washington, MI 48094

Monitor hookup

• Direct video hookup schematic and directions for ZX80 and ZX81 from:

Heuristics
25 Shute Path
Newton, MA 02159

I/O Boards

• 3 port I/O P.C. board, 8255 PIA chip, 16K RAM connector, output connector, with extra board space for A/O converter, relays, clock chip, calculator chip, etc. P.C. Board wired, $45; add-ons extra. Software and specs included. Order from:

Professional Electronics
109 Chesney Lane
Columbia, SC 29209

• RAMPORT, an integrated expansion system for the ZX81

Analogue and digital I/O ports and 16K RAM expansion on one card allowing long programs to be written for controlling motors, solenoids, etc. with appropriate interfacing. Write for details to:

Componedex Ltd.
P.O. Box 33
Leighton Buzzard LU7 7UX
Bedfordshire
United Kingdom

Cassette tape

• Manufacturer of blank load magnetic tape cassettes and certified digital cassettes. Duplicators of High Speed Magnetic Tape. Custom manufacture and duplicate your products and programs. For details write:

Custom Cassette Cartridge Co.
6228 W. Oakton St.
Morton Grove, IL 60053

Games

• *Moving Billboard* Billboard moves sideways from left to right. Display capacity is all of your RAM, 8K ROM only. $1.95 for list. *Horse Race*, 8K ROM $1.95. Both will run on 1K RAM. SASE for catalog of goodies to:

EZRA GROUP II
PO Box 5222
San Diego, CA 92105

• Four games for the 4K ROM, 1K RAM ZX80, $5.00 to:

Aaron Andrews
PO Box 602
Chewelah, WA 99109

• *Adventure C*; a space adventure game. Your space ship is caught by the graviton beam of an alien cruiser and you must escape. The hazards are great, but you have computer rooms, force shields, and lazer guns. £9

ZXCHESS II for the ZX81; Written entirely in machine code; 16K RAM; 7 levels; graphics display of entire board. The computer will even recommend a move for you if you want. £15

Artic Computing
396 James Reckitt Avenue
Hull, North Humberside
United Kingdom

In U.S.

Softsync, Inc.
PO Box 480
Murray Hill Station
New York, NY 10156

And

Gladstone Electronics
901 Fuhrmann Blvd.
Buffalo, NY 14203

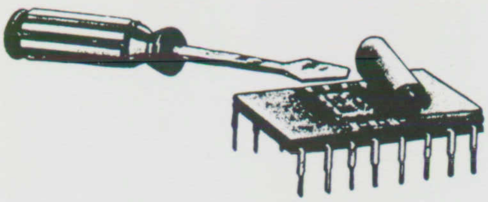
Software Publishers

• Dutch/German scientific and software publishers want to publish and sell your software for the ZX80/81 1K-16K on the European continent. Licenses for already published materials. For details write to:

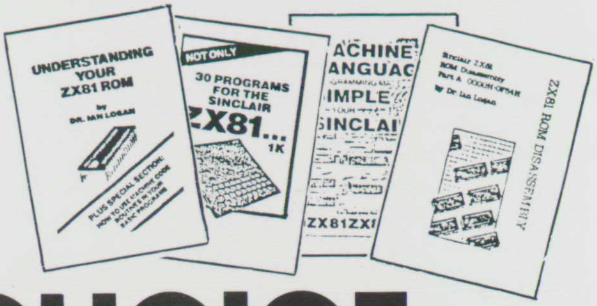
WISSENSCHAFTLICHE EDITION
H. J. Lehmann
van Teylingenlaan 10
1701 AB Heerhugowaard
The Netherlands

FOR ZX81 INFORMATION

DO IT THE HARD WAY ⇦ ⇨ **DO IT OUR WAY**



OR



IT'S YOUR CHOICE:

Not Only 30 Programs ZX81:1K

The ZX 81/1K is more powerful than you ever imagined – this book of over 30 programs, all designed for the unexpanded ZX 81/1K, will show you the amazing range of possibilities, from games, through to educational and mathematical applications, right through to exciting displays of 'Artificial Intelligence'.

Programs include Blackjack, Roulette, Star Wars, Breakout, Memory Left, Mini Adventure, 1K Draughts, all within 1K, and much, much more!!

Each program is explained, with programming hints throughout, including space saving techniques, PEEK and POKE explained, and more!!

US $14.95 UK £6.95

Machine Language Made Simple for your Sinclair

Learn how to program in the ZX 81's own language, Z80 machine language, and get more power from your Sinclair: FASTER RUNNING PROGRAMS (typically 20 times faster than BASIC!), MORE PROGRAM IN LESS SPACE, and COMPLETE CONTROL!

Absolutely essential for the programmer that is interested in going beyond the confines of BASIC, and maximising the potential of the ZX 81.

US $19.50 UK £8.95

Understanding Your ZX81 ROM

In this book Dr. Logan gives a complete overview of Z80 machine language using the ZX 81 monitor program as an example. Dr. Logan explains the structure of the ZX 81 ROM, its peculiarities, and how you can use the ZX 81 ROM routines for your own purposes.

A special section shows you how to use machine code routines in your own BASIC programs! Complete with example programs, reference tables, etc.

US $19.50 UK £8.95

ZX81 ROM Disassembly *Part A

This is the book for the programmer that needs those complete answers about the ZX 81 ROM.

Dr. Logan has examined all routines in the ROM and here he comments on each one. This book is a must for the experienced programmer.

Part A covers all ROM locations from 0000H to 0F54H, and includes all functions except for the routines used in the floating point calculator. Part B (in preparation) covers locations 0F55H to 1DFFH and includes all routines used in the ZX 81 floating point calculator.

US $14.95 £7.00

Please add US $1.00 (UK 50P) to cover post and packing.

Melbourne House Publishers

Melbourne House Publishers is the world's largest publishers of books and software for the Sinclair.
The above titles are all available from the following distributors:

Melbourne House Publishers
131 Trafalgar Road
Greenwich London SE10
United Kingdom

Softsync Inc.
P.O. Box 480
Murray Hill Station
N.Y. 10156

Gladstone Electronics
1736 Avenue Road
Toronto M5M 3Y7
Canada

Gladstone Electronics
901 Fuhrmann Blvd
Buffalo
N.Y. 14203

Make the Most of Your ZX81 or 80

The ZX81 Companion

The ZX81 Companion by Bob Maunder follows the same format as the popular **ZX80 Companion**. The book assists ZX81 users in four application areas: graphics, information retrieval, education and games. The book includes scores of fully documented listings of short routines as well as complete programs. For the serious user, the book also includes a disassembled listing of the ZX81 ROM Monitor.

MUSE reviewed the book and said, "Bob Maunder's **ZX80 Companion** was rightly recognized to be one of the best books published on progressive use of Sinclair's first micro. This is likely to gain a similar reputation. In its 130 pages, his attempt to show meaningful uses of the machine is brilliantly successful."

"The book has four sections with the author exploring in turn interactive graphics (gaming), information retrieval, educational computing, and the ZX81 monitor. In each case the exploration is thoughtfully written, detailed, and illustrated with meaningful programs. The educational section is the same—Bob Maunder is a teacher—and here we find sensible ideas tips, warnings and programs too."

Softbound, 5 1/2 x 8", 132 pages, $8.95.

Getting Acquainted With Your ZX81

This book is aimed at helping the newcomer make most effective use of his ZX81. As you work your way through it, your program library will grow (more than 70 programs) along with your understanding of Basic.

The book is chock full of games such as *Checkers* which draws the entire board on the screen. Other games include Alien Imploders, Blastermind, Moon Lander, Breakout, Digital Clock, Roller-Ball, Derby Day, and Star Burst.

But the book is not all games. It describes the use of PLOT and UNPLOT SCROLL, arrays, TAB, PRINT AT, INKEY$, random numbers and PEEK and POKE. You'll find programs to print cascading sine waves, tables and graphs; to solve quadratic equations; to sort data; to compute interest and much more.

Softbound, 5 1/2 x 8", 120 pages $8.95.

The Gateway Guide to the ZX81 and ZX80

The Gateway Guide to the ZX81 and ZX80 by Mark Charlton contains more than 70 fully documented and explained programs for the ZX81 (or 8K ZX80). The book is a "doing book," rather than a reading one and the author encourages the reader to try things out as he goes. The book starts at a low level and assumes the ZX80 or ZX81 is the reader's first computer. However by the end, the reader will have become quite proficient.

The majority of programs in the books were written deliberately to make them easily convertible from machine to machine (ZX81, 4K ZX80 or 1K ZX80) so no matter which you have, you'll find many programs which you can run right away.

The book describes each function and statement in turn, illustrates it in a demonstration routine or program and then combines it with previously discussed material.

Softbound, 5 1/2 x 8", 172 pages, $8.95.

Computers For Kids, Sinclair Edition

Computers For Kids, by Sally Larsen is the fourth book in this highly successful series. (Previous editions have been released for TRS-80, Apple and Atari computers.) Written expressly for youngsters ages 8 to 13, the book requires no previous knowledge of algebra, variables or computers. Armed with a ZX81 and this book, a child will be able to write programs in less than an hour. A section is included for parents and teachers.

The book starts with a patient explanation of how to use the Sinclair, graduates to flow charts, and simple print programs. The twelve easy-to-read chapters go through loops, graphics and show other programming concepts, and show in a painless way how to make the computer do what you want.

Donald T. Piele, Professor of Mathematics at the University of Wisconsin-Parkside says, "**Computers For Kids** is the best material available for introducing students to their new computer. It is a perfect tool for teachers who are learning about computers and programming **with** their students. Highly recommended."

Softbound, 8 1/2 x 11", 56 pages, $3.95.

Order From

creative computing

Dept. Z0223,
Room 458, One Park Ave., New York, NY 10016

Cash Orders Send payment* plus $2.00 postage and handling to the above address.

Credit Card and Phone Orders (Minimum $10)—Charge your American Express, Visa or MasterCard account. Phone *24 HOURS TOLL FREE* (800) 345-8112. In PA only (800) 662-2444.
By MAIL: Include credit card name, number, expiration date.

*Residents of CA, CO, DC, FL, IL, MA, MI, MO, NJ, NY State, OH, SC, TN, VT, add applicable sales tax.